

V. MAGYAR SZÁMÍTÓGÉPES GRAFIKA ÉS GEOMETRIA KONFERENCIA

Budapest
2010. JANUÁR 26-27.



Szerkesztette:
Szirmay-Kalos László
Renner Gábor



Előszó

A Magyar Számítógépes Grafika és Geometria Konferencia sorozat immár az ötödik rendezvényéhez érkezett, köszönhetően a Neumann János Számítógép-tudományi Társaság (NJSZT) támogatásának, a Számítógépes Grafika és Geometria Szakosztály (NJSZT-GRAFGEO) lelkesedésének, a SZTAKI és a BME Irányítástechnika és Informatika Tanszéke segítségének, és nem utolsósorban a számítógépes grafika és geometria magyar és magyar kötődésű művelőinek. Az elmúlt konferenciák bevált szervezési módszereit alkalmaztuk most is, reménykedve abban, hogy ez az ötödik, már akár jubileuminak is tekinthető alkalom a korábbi konferenciákhoz hasonlóan sikeres lesz.

Ebben az évben 32 cikket válogattunk be a végleges programba, amely minden korábbinál nagyobb szám. A számítógépes grafika és geometria tehát fejlődik hazánkban, amihez egy kicsit talán ez a konferenciasorozat is hozzájárul. A cikkek áttekintést adnak a hazai kutatócsoportok érdeklődési területeiről, eredményeiről, sőt még részben azok külföldi kapcsolatairól is. A hagyományos témák, mint a geometriai modellezés, valósidejű képszintézis, 3D objektumrekonstrukció mellett egyre erősebbé válik a GPU-k szimulációs és képfeldolgozási alkalmazása is.

A kiadványt Umenhoffer Tamás készítette elő nyomtatásra, a címlapképet Balambér Dávid alkotta meg. Önzetlen munkájukért és a felsorolt szervezetek támogatásáért a konferencia szervezői ezúton fejezik ki a köszönetüket.

Budapest, 2010. január 15.

Szirmay-Kalos László és Renner Gábor

Tartalomjegyzék

Balambér Dávid Cover Story: How did we prepare the cover images?.....	1
Várady Tamás Advanced Techniques to Create CAD Models from Measured Data.....	2

Valós idejű képszintézis I.

Áfra Attila Interactive Out-of-Core Ray Casting of Massive Triangular Models with Voxel-Based LODs.....	4
Liktor Gábor, Dachsbacher Carsten Real-Time Volumetric Caustics with Projected Light Beams.....	12
Ralovich Kristóf, Magdics Milán Recursive Ray Tracing in Geometry Shader.....	19

Valós idejű képszintézis II.

Huszár Tamás, Szécsi László Real-time foam simulation on the GPU.....	27
Umenhoffer Tamás Graphics Techniques in the Turing Game Project.....	33
Heisenberger Viktor, Szécsi László Shadow map filtering for rendering volumetric phenomena.....	40
Szécsi László, Szirmay-Kalos László, Kurt Murat Adaptive Sampling with Error Control.....	47

Geometriai modellezés

Szilvási-Nagy Márta Construction of circular splats on analytic surfaces.....	55
Róth Ágoston, Juhász Imre Interpolation with cyclic curves and surfaces.....	58
Salvi Péter, Várady Tamás Constrained Fairing of Freeform Surfaces.....	65
Gyurecz György, Renner Gábor Evaluating and Correcting the Reflection Characteristics of Surfaces.....	73
Hajdu András, Tóth Tamás Approximating non-metrical distances in 3D.....	81

3D rekonstrukció

Hajder Levente L2-Optimal Weak-Perspective Camera Calibration.....	89
Kovács Tibor Accuracy Analysis of an Enhanced Peak Detector.....	97
Pernek Ákos, Hajder Levente Accurate 3D Reconstruction and Camera Auto-Calibration.....	102
Jankó Zsolt, Pons Jean-Philippe Image-Based Texturing of Dynamic 3-D Models.....	110
Molnár József, Csetverikov Dmitrij Másodfokú közelítés implicit felületek síkbeli leképezésére.....	118

Orvosi képfeldolgozás

Soumelidis Alexandros, Fazekas Zoltán, Pap Margit, Schipp Ferenc Discrete orthogonality of Zernike functions and its relevance to corneal topography.....	125
Tomán H., Hajdu András, Szakács J., Hornyik D., Csutak A., Pető Tibor Thickness-based binary morphological improvement of distorted digital line intersections	133
Csász I., Csutak A., Pető Tibor, Hajdu András Well fitting curve models with few parameters for vascular systems on retinal images.....	140
Kriston András, Bekes György, Ruskó László, Fidrich Márta Atlas-based abdomen segmentation.....	145
Ungi Tamás, Kriston András, Blaskovics Tamás, Fidrich Márta Semi-automatic framework for anatomical liver segment separation.....	149

Valós idejű szimuláció GPU-n

Srp Ágoston, Vajda Ferenc High-Speed Implementation of Bleeding Detection in Capsule Endoscopy Images using GPGPU Computing.....	155
Magdics Milán, Tóth Balázs, Csendesi Ádám Iterative 3D Reconstruction with Scatter Compensation for PET-CT on the GPU.....	159
Umenhoffer Tamás, Szirmay-Kalos László Volumetric Ambient Occlusion for Volumetric Models.....	169
Tóth Balázs, Magdics Milán Monte Carlo Radiative Transport on the GPU.....	177
Csébfalvi Balázs, Domonkos Balázs, Vad Viktor Recent Results on Optimal Regular Volume Sampling.....	185
Tukora Balázs, Szalay Tibor LDNI alapú söpört térfogat generálása anyageltávolítás jellegű gépészeti szimulációhoz....	193

Gépi látás és videó

Takács Barnabás, Szijártó Gábor, Komáromy-Mészáros Gergely, Hanák Dávid, Dobson Levente, Kömíves Zoltán Immersive Interactive Film & Real-Time Computer Graphics.....	198
Prohászka Zoltán Matching Image Details with the Self Affine Feature Transform.....	206
Helfenbein Tamás, Vajda Ferenc Motion Detection Using Low Resolution Video Sequences.....	214
Fazekas Zoltán, Gáspár Péter, Soumelidis Alexandros Experiments towards an on-vehicle omni-directional camera-based road-safety assessment system.....	220

Cover Story: How did we prepare the cover images?

Dávid Balambér (BME IIT)

Abstract

This paper explains the story of the creation of the cover images that show caustic patterns generated by simple metallic diffusors and glass lens, demonstrating that very simple mathematical models can lead to complex and beautiful images. The rendering program was developed as a homework assignment of the Computer Graphics and Image Processing course at BME.

1. The scene

The caustics on the front and back covers of this proceedings are generated by a simple ray tracing simulator program written in C++. The rendered scene consists of an isotropic point light source, a plane as the illuminated surface, and two quadratic surfaces realizing a lamp: a golden elliptical paraboloid as lamp-shell, and a glass ellipsoid as lens. The light source is inside the paraboloid, and the ellipsoid seals the lamp-shell paraboloid, so that most of the light goes through the ellipsoid (See Figure 1 for side-view).

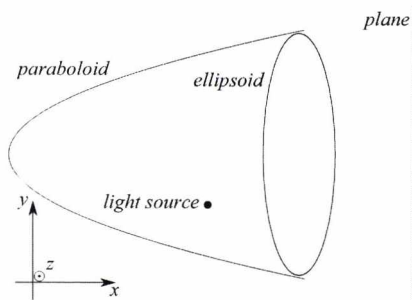


Figure 1: The scene.

2. Rendering

The simulator is the implementation of a Monte Carlo photon tracing global illumination algorithm. It launches a number of white colored rays from the light source point in uniformly distributed directions, simulating the isotropic light source. Rays are intersected with the scene objects, and the target object is determined by the nearest intersection on the ray path. Rays hitting the golden paraboloid shell are reflected, altering the ray's red, green and blue color values ac-

ording to the Fresnel function computed from the refractive indices and extinction coefficients of gold. As for rays that hit the ellipsoid lens, rays are both reflected and refracted (if refraction is possible, i.e. there is no total internal reflection) according to the refraction index and extinction coefficient of glass (chosen the same for all colors, i.e. dispersion is not taken into account for simplicity). The whole procedure is repeated recursively until the rays hit the plane, adding their red, green and blue color values to the pixel that represents the corresponding square surface elements on the plane. The last step is tone mapping that scales the accumulated color values to displayable values.

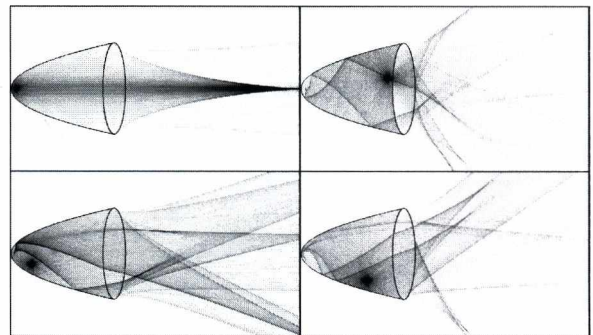


Figure 2: Two dimensional light density maps, demonstrating how caustics change when modifying the light source's position.

The final images were obtained by experimenting with different paraboloid and lens parameters, and also with different light source positions (Figure 2). For each of the images, we used 10^{10} Monte Carlo sample rays traced from the source.

Advanced Techniques to Create CAD Models from Measured Data

Tamás Várady¹

¹ Geomagic, Inc., varady@geomagic.com

Abstract

Digital shape reconstruction is the process of creating digital models from physical parts represented by 3D point clouds. The ideal process is expected to provide a boundary representation that is likely to be identical or similar to the original design intent of the object, and requires minimal user assistance. In this talk alternative state-of-the-art approaches are discussed, where emphasis is put on automatic methods (i) to create complete and consistent topological structures over polygonal meshes; and (ii) extract accurate and properly aligned surface features that yield complete, trimmed CAD models with fillets and corner patches. Problems and recommended solutions will be presented through case studies using industrial parts.

Geomagic is a leading technology provider of 3D software for creating digital models. The talk will also briefly introduce the company's main products Geomagic Studio for digital shape reconstruction and Geomagic Qualify for inspecting industrial parts.

Keywords: Digital shape sampling and processing, reverse engineering, segmentation, functional decomposition, redesign over meshes.

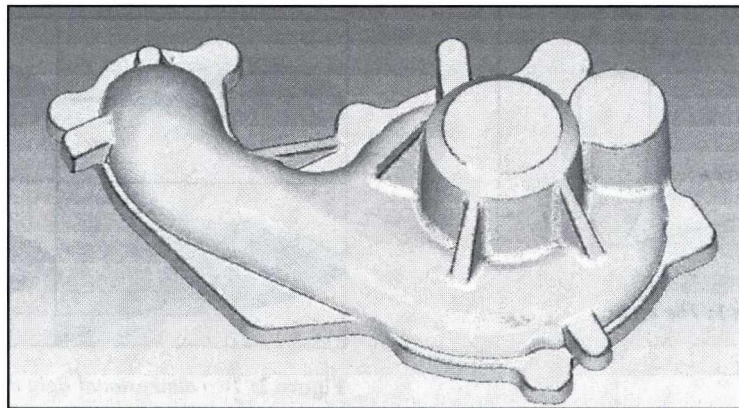


Figure 1: Polygonal mesh

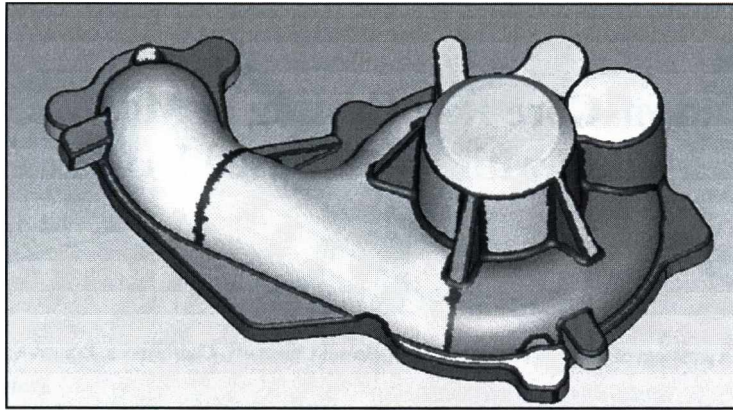


Figure 2: Automatic segmentation

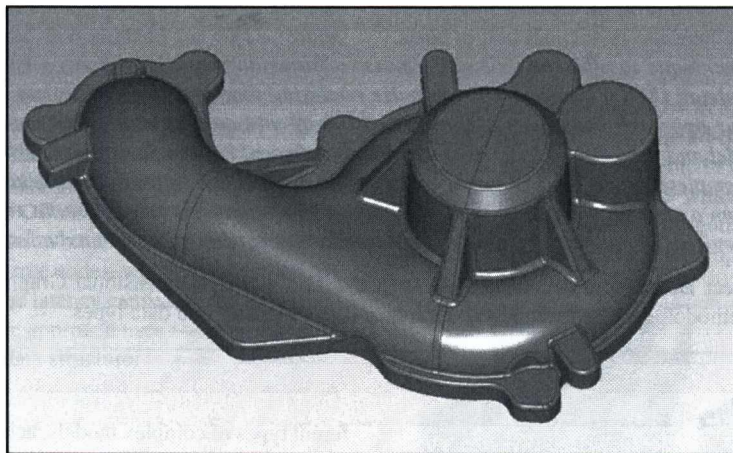


Figure 3: Final CAD model

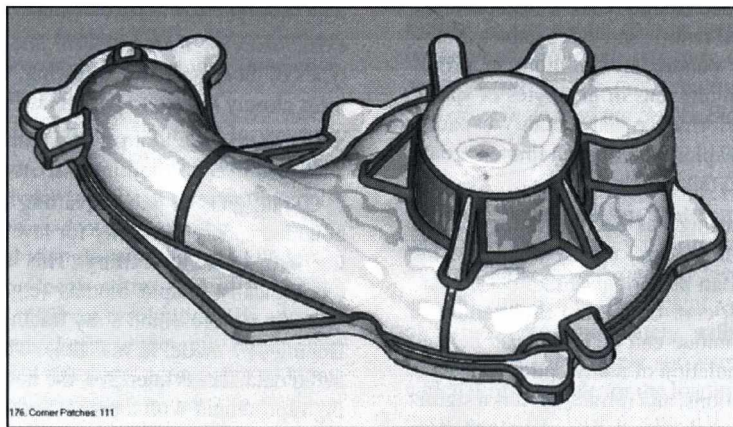


Figure 4: Deviation map

Interactive Out-of-Core Ray Casting of Massive Triangular Models with Voxel-Based LODs

Attila T. Áfra¹

¹ Department of Computer Science, Babeş-Bolyai University, Cluj-Napoca, Romania

Abstract

We propose a new technique to efficiently visualize massive triangular models with ray casting. We employ a voxel-based level-of-detail (LOD) framework to minimize rendering time and required system memory. In a pre-processing phase, our approach constructs a compressed out-of-core data structure that contains the original primitives of the model and the LOD voxels, organized into an efficient kd-tree. During rendering, data is loaded asynchronously to completely hide the I/O latency. Only a fraction of the entire data structure is necessary to render the model from a specific viewpoint. With our approach, we are able to explore, in real-time, data sets consisting of even hundreds of millions of triangles on a commodity notebook PC with a dual-core CPU.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Raytracing I.3.6 [Methodology and Techniques]: Graphics data structures and data types

1. Introduction

The real-time visualization and inspection of highly complex 3D models is required in many scientific, engineering, and entertainment domains. Examples of such domains include, among others, computer-aided design (CAD), 3D scanning, numerical simulation, virtual reality, and video games. Many massive models consist of hundreds of millions of primitives (e.g., triangles), occupying tens of gigabytes of space. Even though processor performance and memory capacity are rapidly increasing, the exploration of such immense data sets can still be problematic on a single commodity PC.

Real-time ray tracing/casting has become a very active research area in the past few years, mostly thanks to the emergence of affordable, high performance parallel processor architectures like multi-core CPUs and programmable GPUs. This rendering technique can be easily parallelized and enables the precise simulation of many optical phenomena such as shadows, reflections, and refraction. It is a significantly more versatile approach than rasterization, the most popular real-time rendering algorithm.

This paper presents a new real-time massive model rendering method based on ray casting. Several testing examples demonstrate that our approach works effectively for dif-

ferent types of complex models, achieving interactive speeds on a mainstream dual-core notebook PC.

2. Previous work

Although several massive model visualization approaches exist, many of them perform adequately only for specific types of models. In the following, we highlight the methods most closely related to our work. For a more comprehensive overview of this topic, we refer the reader to the synthesis lecture by Yoon *et al*¹⁰.

QSplat^{7,6} is a point splatting technique which uses a bounded sphere hierarchy for level-of-detail (LOD) rendering with visibility culling. This approach works only for topologically simple models (e.g., laser-scanned statues). Wald *et al*.⁹ presented a ray tracing method to visualize the Boeing 777 model in real-time. The I/O is asynchronous to avoid data access latencies, the not-yet-loaded geometry being represented with a small number of volumetric proxies. This is not a full LOD solution, thus, the ray traversal depth and the working set size are not reduced. The Quick-VDR algorithm¹² uses a clustered hierarchy of progressive meshes (CHPM) for out-of-core rendering and occlusion culling. Without geomorphing, popping artifacts may occur while



Figure 1: Massive models rendered with the proposed technique: Power Plant (12M triangles), Lucy (28M triangles), and Mandelbulb (354M triangles).

switching between different LODs. Far Voxels³ employs a LOD framework based on cubical view-dependent voxels, uses asynchronous I/O, and is optimized for GPUs. Since it renders voxels by splatting, the artifacts due to insufficient available data are more distracting if the model is less smooth. The ray tracing based algorithm proposed by Yoon *et al.*¹¹ uses drastic simplifications, called R-LODs, to improve rendering performance and minimize the amount of required memory. The R-LODs are tightly integrated with a kd-tree used as an acceleration structure, and they consist of simple planes bounded by tree nodes, which have limited approximation capability. The latency caused by the data loading is not hidden, since the approach uses *memory mapping* to access the out-of-core data structure.

3. Method overview

Our approach is based on ray casting, a subset of ray tracing. While we cast only primary rays, the proposed method could be extended to recursive ray tracing. We have implemented the renderer entirely on the CPU, but the method can be efficiently adapted also to modern GPUs. By exploiting the power of multi-core processor architectures, interactive rendering speeds can be attained.

The main goal is to seamlessly explore massive models, consisting of possibly hundreds of millions of triangles, on a single commodity PC. In order to achieve this, we first construct an optimized *compressed* out-of-core data structure, which contains the original triangles and several LOD levels consisting of voxels. These levels correspond to simplified versions of the data set at different resolutions. We approximate the model with voxels, because they are suitable for geometry with very complex topology, comprising of many detailed, loosely connected, interweaving parts (e.g., pipes and wires).

All primitives are organized into a *kd-tree*, a simple, efficient, and widely employed acceleration structure for ray tracing⁴. This is essentially a binary space partitioning (BSP) tree in which every non-leaf node divides the space into two

subspaces with an axis-aligned plane, and the leaf nodes contain references to primitives. This data structure can be used to find the closest intersection of a ray with the primitives in $O(\log n)$ time.

The voxels are bounded by tree nodes, thus, the LOD data is tightly integrated with the kd-tree. Note that this way the voxels do not have to be cubic in shape. The entire structure is decomposed into *treelets*² which are grouped into equally sized *blocks*. A lossless data compression algorithm is applied to store the blocks in a compact form.

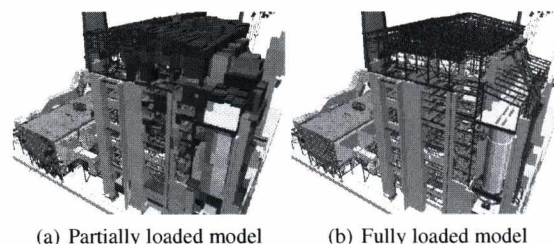


Figure 2: The model data is loaded asynchronously in the background. Although this may cause temporary blocking artifacts, the exploration remains fluid. This figure shows this effect in case of the Power Plant model.

Thanks to the hierarchical LOD mechanism, it is possible to render huge data sets that cannot be completely loaded into the system memory. During rendering, we load the necessary details *asynchronously* (see Figure 2), thus, there is no stuttering due to insufficient available data. Also, the exploration starts immediately, without any loading time. We employ a custom, purely software-based *memory manager*, which is responsible for the loading of the blocks required by the renderer. We have designed the out-of-core data structure with a software solution in mind, and as a consequence, the software address translation overhead is greatly reduced. On the other hand, the application of a custom memory management mechanism enables us to perform superior block caching and on-the-fly decompression.

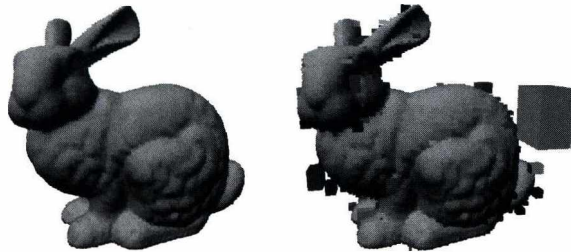
The proposed rendering method is an extension of the efficient ray traversal algorithm introduced by Wald⁸, therefore many known optimizations (like *ray packets*¹) can be adapted to our approach. It is important to note that by using LODs, significantly higher frame rates can be achieved with minimal loss of image quality, because ray traversals are less deep, intersections with voxels are implicit, and memory accesses are more coherent. Furthermore, the LOD framework can also reduce the amount of aliasing artifacts, especially in case of highly tessellated models.

4. Construction

In a preprocessing stage, we construct the optimized out-of-core data structure for the mesh that we would like to visualize interactively. We assume that the original model is stored as a *triangle soup* (i.e., a simple list of triangles without shared vertex data).

4.1. Kd-tree

The main part of the construction process is the building of the kd-tree, which consists of the recursive spatial partitioning of the scene using axis-aligned planes. The inner nodes of a kd-tree contain an axis-aligned splitting plane, and the leaf nodes refer to a set of triangles. Since the size of the data set may exceed the amount of available system memory, we employ out-of-core algorithms where necessary. We first determine the axis-aligned bounding box of the model, then we proceed with the building of the kd-tree nodes, in depth-first order.



(a) With empty space cutting (b) Without empty space cutting

Figure 3: The highest resolution voxel-based LOD level of the Stanford Bunny model constructed with and without empty space cutting. The image on the right clearly shows that using solely the surface area heuristic does not result in satisfactory approximation quality.

If all primitives belonging to a node can fit into the memory, we determine the splitting plane using the well-known *surface area heuristic* (SAH), which produces high quality results⁴. Otherwise, we split the node in the *middle*, in an out-of-core fashion. This operation takes as input a triangle list stored in a file stream, and produces two new streams

corresponding to the children of the respective node. The sifting of the triangles into two sublists can be achieved in only a single sweep over the input file stream. This is an important property of the method, because disk I/O operations are very time-consuming. Although splitting in the middle is almost always inferior to the surface area heuristic, only a relatively small amount of nodes close to the root are required to be built this way. Thus, the rendering performance is *not* affected in a significant way.

A kd-tree node may contain a LOD voxel, which fills entirely the cell determined by the node in question. If we use only the surface area heuristic (or middle splitting), the resulting voxels will not provide an accurate enough approximation of the original geometry (see Figure 3). In order to solve this problem, we additionally employ an aggressive *empty space cutting* strategy. Before selecting a splitting plane using one of the mentioned methods or making the node a leaf, we check whether we can create an empty child node that has a volume respective to the original cell larger than a specified threshold (we used values between 10-20%). This approach not only improves LOD quality, but also increases ray tracing performance⁵.

4.2. Voxels

A voxel roughly approximates the original primitives stored in the subtree of a kd-tree node and holds material attributes used for shading during the rendering process. The extent of a voxel stored in a node is implicitly defined by the node itself. In our implementation, we use two types of material attributes: *normal* and *color*. If the geometry is simple enough, a single set of attributes, called a *voxel sample*, is sufficient. Otherwise, we store separate samples for the six sides of the voxel.

Creating and storing voxels in every kd-tree node is very expensive, therefore, we compute voxels only for a subset of the nodes. Since a kd-tree node partitions space only in one dimension, it is adequate to select every third node on the path from the root to a leaf.

If we have to build a node out-of-core, we combine the computation of the voxel samples with the triangle sifting process. We rasterize the triangles onto six image planes corresponding to the sides of the respective voxel. In order to take into account occlusion, we perform depth testing. After processing all triangles, we average the unoccluded samples. When building a subtree in-core, we employ adaptive Monte Carlo ray sampling. This can be implemented very efficiently by using the prebuilt kd-subtree rooted at the current node to accelerate the ray traversal.

We encode the normals in a 30-bit quantized form (10 bits per component) and the colors in standard 24-bit RGB format. We pad both attributes to 32 bits. Using this encoding, the size of a 1-sample voxel is 8 bytes and a 6-sample voxel is 48 bytes.

4.3. Blocks

In order to render an image of a massive model, usually only a fraction of the kd-tree nodes must be accessed. We exploit this property by using an out-of-core representation that enables us to efficiently traverse the kd-tree without keeping the entire data structure in memory.

The proposed format is block-based, which allows fast loading and simple memory management. In our implementation, the blocks have equal size of 64 KB. In fact, these blocks are similar to memory *pages* that are managed by the CPU and the operating system. By using a custom memory manager, the block size can differ from the system page size, which is commonly 4 KB. Another important benefit is the possibility to store the blocks on the hard disk in a compressed form and decompress them in real-time. This not only reduces storage requirements, but may also increase loading performance, because less disk operations are necessary. On-the-fly decompression should not degrade significantly the rendering performance, thus we apply a simple and very fast LZ77 family algorithm (e.g., LZO and QuickLZ). The decompression speed of such methods can be even higher than the speed of simple memory-to-memory copying on modern processor architectures.

4.4. Treelets

We decompose the kd-tree and all related data into treelets which are small subtrees with a fixed maximum height of 3. We store these in blocks in *breadth-first* order, packing many consecutive entire treelets into a single block. Since there are voxels only in every third tree level, treelets that have roots at the same depth constitute a LOD level. According to this simple treelet layout, the LOD levels are stored continuously and consecutively, starting with the lowest level-of-detail.

A treelet consists of different types of kd-tree nodes, which may refer to additional data. To encode these in a highly compact way, variable size treelet nodes are used. The size of a node is either 8 or 16 bytes, so a cache-friendly memory alignment is possible. Also, the bit representation is carefully designed to minimize the number of operations required to unpack the node data.

The ordering of the nodes is strict, and follows the runtime memory access pattern: a node must be stored *before* its left child which must be stored *before* the right one.

4.4.1. Inner nodes without a voxel

Most of the treelet nodes are inner nodes *without* a voxel, which have children located in the same treelet. In addition to the type of the node and the splitting plane, child addresses are also encoded. The amount of required bits is reduced by storing the offset of the left child from the parent and the offset of the right child from the left one. These offsets are strictly *positive* and do not exceed the block size.

A significant amount of the child nodes are *empty leaves*, which should not be stored to save memory. An efficient solution is to encode bits in the inner nodes that indicate whether a child node is empty or not.

All in all, the node data can be fit into 8 bytes. Note that this is also the size of the state of the art in-core node representation proposed by Wald⁸. The complete bit structure is summed up in Table 1.

4.4.2. Inner nodes with a voxel

An inner node *with* a voxel points to roots of other treelets that are located in a different block. Because of this, the aforementioned node offsets are not sufficient to address the child nodes, thus, a 32-bit *block ID* is also encoded. This way, the size of the addressable memory space increases to 256 TB. In order to store only one block ID, the two child treelets are restricted to be in the same block.

During rendering, voxels are significantly less frequently accessed than nodes. Thus, instead of the non-cache-efficient interleaving of the voxels with the nodes, voxels are stored separately, after all the nodes of the treelet, and the nodes contain a small offset to the voxel data.

Bits	Value
0–1	split axis
2	left child empty flag
3–15	left child offset from this node
16	right child empty flag
17–21	right child offset from left child
22	voxel flag
23–29	voxel offset from this node (only w/ voxel)
30	(not used)
31	inner node flag (1)
32–63	split position
64–95	enclosing sphere radius (only w/ voxel)
96–127	children block ID (only w/ voxel)

Table 1: The bit structure of inner nodes. All offsets are multiples of 8. Some fields are only necessary if the node has a voxel. The total size is either 8 or 16 bytes.

When the ray tracer accesses a node which contains a voxel, it has to compute a LOD error value to decide whether to continue with the traversal or not. This error value depends, among others, on the size of the voxel, the on-the-fly calculation of which is costly. Therefore, the size is stored in the tree, encoded as the radius of the sphere enclosing the respective voxel. Since this radius is a frequently accessed value, it is part of the node data structure.

As listed in Table 1, the size of the inner nodes with a voxel (excluding the voxel data) is twice (i.e., 16 bytes) the size of the normal inner nodes. Although this a significant increase, the total size of the kd-tree is *not* affected dramatically, because only a fraction of the nodes are of this type.

4.4.3. Leaf nodes

The leaf nodes in a kd-tree contain a subset of the original triangles, and, usually, many of these triangles are shared by multiple nodes. Because of this, it is not efficient to store the triangles themselves in each node. In case of simple in-core kd-tree representations, it is common practice to store each triangle only once in a global array, and create triangle lists that belong to the leaf nodes which contain pointers to the triangles. The nodes, triangles lists, and triangles are put into different buffers.

For our out-of-core data structure, we take a similar approach. Instead of global buffers, we create *triangle blocks*, each divided into a *triangle list section*, a *triangle section*, and a *vertex section*. These blocks represent the highest level-of-detail, and we output them *after* the other blocks.

The triangle list section contains a series of 16-bit offsets that point to a 12-byte triangle structure (with color). Usually, the model is a triangle mesh, so a vertex may belong to more than one triangle. To further minimize the storage requirements, the unique vertices are stored in the vertex section, and the triangle structures contain three offsets to one of these vertices. The shared vertices are determined with *vertex hashing*. If the vertices have only a position attribute, they are encoded in 12 bytes.

A leaf node is a simple 8 byte structure (see Table 2), which contains a reference to a triangle list and the number of triangles in that list. The list reference is expressed as a block ID and an offset from the beginning of the block.

Bits	Value
0–15	triangle list offset
16–30	triangle count
31	inner node flag (0)
32–63	triangle block ID

Table 2: The bit structure of leaf nodes. The total size is 8 bytes.

5. Rendering

As previously mentioned, rendering is performed by ray casting, and the out-of-core data structure is accessed through a custom memory manager.

5.1. Ray casting

In order to compute the color of a pixel, we cast a ray from the camera viewpoint through the pixel and determine its nearest intersection with the geometry. Then, we perform shading at the intersection point, while optionally casting secondary rays to visualize shadows, reflections, or other

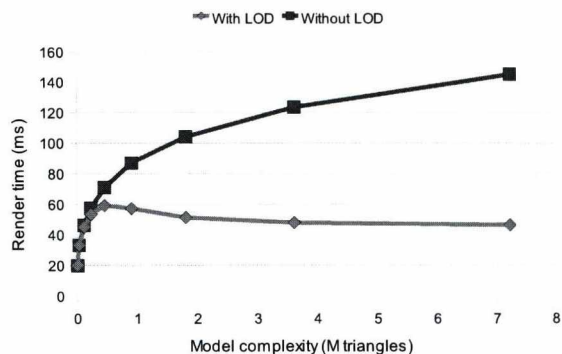


Figure 4: The scaling of the ray casting performance with the model complexity. We have created simplified versions of the Asian Dragon model (7M triangles) and rendered them with our method from the same viewpoint. We have measured the render time of a single frame with and without using LOD. Notice that with LOD, the performance becomes nearly constant after a certain model complexity.

light phenomena. Ray casting consists of two main operations: recursive acceleration structure traversal and primitive intersection. In our case, the acceleration structure corresponds to the out-of-core kd-tree, while the primitives are of two types: triangles and voxels.

We extend the traversal algorithm proposed by Wald⁸ to support voxel LODs and asynchronous loading. During traversal, we additionally maintain the reference to the last encountered voxel on the current path. When we reach an inner node with a voxel, we update this reference and compute a LOD error value. If the error does not exceed a specified threshold, we intersect the ray with the voxel and stop the traversal. It may happen that we cannot continue with the traversal from the current node, because the required data (child node or triangle list) is not yet loaded into memory. In this case, we return the intersection with the last encountered voxel, clipped to the bounds of the current node.

We employ a simple and low cost projected screen-space error metric also used by Yoon *et al*¹¹. The error threshold is expressed as a *pixels of error* (PoE) value, which is the maximum allowed screen-space area of a projected voxel. To simplify the calculations, we approximate the voxel with its enclosing sphere and compare its screen-space radius with the threshold corresponding to the PoE. This threshold is simply the radius (\hat{R}_{\max}) of the circle which area is equal to the PoE. The estimated perspective projected screen-space radius (\hat{R}) of a voxel enclosed by a sphere with radius R can be written as:

$$\hat{R} = \lambda \frac{R}{t_{\min}}, \quad \lambda = \frac{w/2}{\tan(\phi/2)}, \quad (1)$$

where t_{\min} is the distance from the viewpoint to the closest ray intersection with the voxel, and λ is a screen-dependent

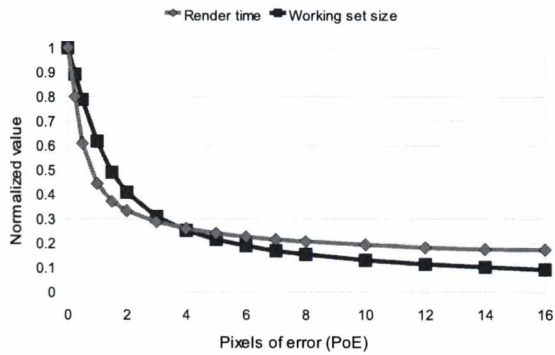


Figure 5: The scaling of the ray casting performance with the pixels of error (PoE). We have rendered the Lucy model with varying PoE values from the same viewpoint and measured the render time and working set size. The values shown in this figure are relative to the maximum measured values.

constant determined by the number of pixels w along the field of view ϕ . Since the voxel is bounded by the node, t_{\min} also corresponds to the intersection with the node. To efficiently check whether the error metric is satisfied, we evaluate the following rearranged inequality:

$$R \leq t_{\min} C, \quad C = \frac{\hat{R}_{\max}}{\lambda}, \quad (2)$$

where C is a global for all rays, which can be precomputed. This way, only a multiplication and a comparison operation must be executed for each encountered voxel.

We shoot rays with normalized direction vectors, therefore, distances to points on a ray (like t_{\min}) are also *ray parameters*. In each ray traversal step, we maintain the intersections of the ray with the current node as a $[t_{\min}, t_{\max}]$ ray parameter interval. This represents the *current ray segment*⁸, which is required to determine which child nodes are intersected by the ray, and to get the closest intersection with a voxel.

Our LOD metric could be also used for several types of secondary rays, including shadow, ambient occlusion, and planar reflection rays. One drawback of this kind of metric is that it works only with secondary rays expressible as linear transformations. Because of this, refraction and non-planar reflection rays are *not* supported¹¹.

To shade a voxel, first we have to get the material attributes at the intersection point. If the voxel has only one sample, the solution is trivial, since the material attributes are constant throughout its surface. But this is not the case if there are separate samples for the six sides of the voxel. We use simple nearest-neighbor filtering, therefore, we have to determine on which side of the voxel is the intersection point. A side can be identified by an axis and a sign (e.g., X+, Z-). The axis of an intersected voxel side cannot be determined implicitly, but the sign is always the opposite of

the sign of the ray direction along the axis of the side. We maintain in each traversal step the axis of the plane through which the ray enters the node. If we traverse the *near child* node, the *entry axis* does not change. However, if we traverse the *far child*, the entry axis becomes the axis of the splitting plane of the parent node.

The proposed ray casting algorithm can be applied not only to single rays but also to ray packets. By tracing multiple rays together, the coherence of the rays and the SIMD instruction set of the processor can be exploited to significantly improve performance. Ray packet tracing combined with a LOD mechanism is especially efficient, because the traversal is usually less deep, so the rays are more coherent.

5.2. Memory management

The renderer accesses all data through a custom memory manager, which operates on the granularity of blocks. Blocks required by the renderer threads are loaded and decompressed into a *fixed-size cache* by one or more separate *fetcher* threads. The cache consists of *cache slots*, and a simple table is used to store the mapping between block IDs and cache slot IDs. We associate with each cache slot a *timestamp*, which indicates when the block stored in the respective slot was last accessed. This information is used to determine which blocks should be evicted from the cache if there are not enough free cache slots to load new blocks.

In order to minimize the overhead of the asynchronous block loading, we do not synchronize the renderer and fetcher threads while rendering a frame. If a renderer thread encounters a reference to a block other than the current one, it requests the address of the block from the memory manager. Then, the manager checks whether the block is loaded into the cache. If so, it *atomically* updates the timestamp of the proper cache slot, and performs address translation by computing and returning the block pointer. Otherwise, it adds an entry to the *request list* of the renderer thread and returns a null pointer.

The fetcher threads continuously load the blocks referenced in a *fetch list* into preallocated cache slots. After loading and decompressing a block, it is not immediately marked as available. As a consequence, the set of available blocks is constant while a frame is being computed.

After finishing the rendering of a frame, first we mark all freshly loaded blocks as available and update a *least recently used* (LRU) list based on the timestamps of the cache slots. Then, we gather the requests from the request lists and assign priorities to them according to the number of affected pixels. We generate block fetch jobs, sort them by their priority, and put them into the fetch list, discarding its previous contents. The priority of a job is the highest priority of all requested data located in the respective block. We limit the length of the fetch list based on the estimated maximum I/O bandwidth. This fetching approach is similar to the one incor-

porated into the Streaming QSplat algorithm⁶. Since blocks must not be removed from the cache while rendering, we *preallocate* cache slots for the blocks to be fetched. If the cache is full, we use the LRU list to evict blocks, starting with the least recently used. In order to avoid cache thrashing, we do not remove blocks that are part of the current working set.

6. Results

We have implemented our method using C++ with SSE intrinsic functions and evaluated its performance on a laptop with an Intel Core 2 Duo T5500 (dual-core, 1.66 GHz, 2 MB L2 cache) processor, 2 GB RAM, and a 7200 RPM hard disk. The system was running Windows 7 64-bit. The code was compiled for 64 bits with Visual C++ 2008 SP1. The data compression was realized using the QuickLZ library.

We have extensively tested the algorithm with several models, but the performance evaluation was mostly restricted to three complex data sets, shown in Figure 1, that are part of different domains: the Power Plant data set (12M triangles) is a complete CAD model of a coal fired power plant; the Lucy data set (28M triangles) is a high resolution laser-scanned statue; the Mandelbulb data set (354M triangles) is a fractal mesh obtained by isosurface extraction.

6.1. Construction

Our prototype implementation of the out-of-core preprocessing stage was designed for simplicity instead of performance. It is single-threaded, thus the construction time in a multi-core or multi-processor environment could be significantly reduced by parallelizing the expensive kd-tree building and the voxel sampling. The construction statistics for the testing models are listed in Table 3.

	Power Plant	Lucy	Mandelbulb
triangles	12M	28M	354M
nodes	76M	278M	903M
leaves	38M	139M	451M
non-empty leaves	22M	62M	180M
avg. tris/NE leaf	3.25	2.54	2.04
voxels	12M	46M	150M
1-sample voxels	3M	42M	91M
1-sample voxel ratio	26%	92%	60%
LOD levels	29	26	26
uncompressed size	1.4 GB	3.2 GB	16 GB
compressed size	0.8 GB	2.2 GB	9.8 GB
compression ratio	55%	68%	61%
build time	55m	84m	698m

Table 3: Construction statistics for the most complex models used for testing: the number of triangles in the model, kd-tree and LOD statistics, the size of the out-of-core data structure, and the build time in minutes.

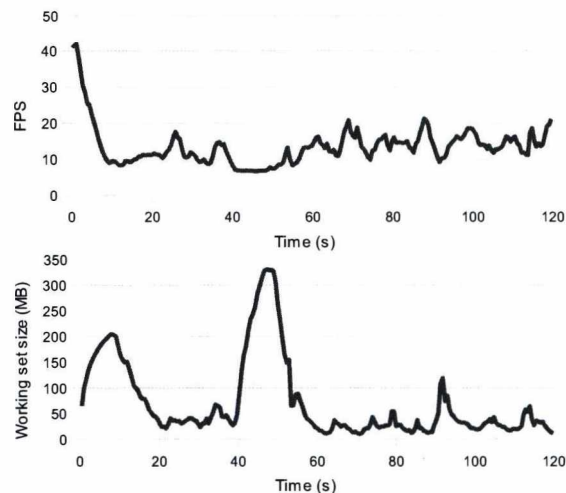


Figure 6: The variation of the rendering speed (13.6 frames per second on average) and the size of the working set (62 MB on average) during a walkthrough of the Mandelbulb model at 640×480 resolution, 1 ray per pixel, and 2 pixels of error.

6.2. Rendering

In order to exploit the coherency of the rays, we employ 4×4 ray packet casting implemented with SSE vector operations. Furthermore, we subdivide the image into *tiles* and distribute them among the renderer threads. We execute a renderer thread on each logical processor, and we achieve almost linear performance scaling with the number of cores.

The scaling of the ray casting performance with the number of triangles in the model is illustrated in Figure 4. Notice that without LOD, the render time increases logarithmically, as expected from employing a kd-tree as an acceleration structure. However, if we enable the use of the voxel-based LODs, the performance becomes nearly *constant* as soon as the maximum projected triangle size drops below the pixel error threshold. Because of this property, the rendering of extremely complex models with our approach can be interactive even on a single desktop or portable computer.

Figure 5 shows that by increasing the PoE value, the render time and the size of the working set can be dramatically reduced, at the expense of image quality. According to our tests, a good trade-off between performance and quality can be achieved with about 2-3 pixels of error.

It can be seen on Figure 6 and Table 4 that the aforementioned complex models can be explored in real-time with our renderer, running on a dual-core notebook CPU. Thanks to the LOD mechanism, the required system memory is also kept within acceptable bounds. We could attain on average approximately 14-20 frames per second at 640×480 resolution, 1 ray per pixel, and 2 pixels of error.

	Power Plant	Lucy	Mandelbulb
average FPS	19.5	15.2	13.6
minimum FPS	7.6	6.4	6.7
maximum FPS	75.9	67.6	41.9
average WS	26 MB	81 MB	70 MB
minimum WS	0 MB	9 MB	10 MB
maximum WS	157 MB	242 MB	331 MB

Table 4: Frame rate (FPS) and working set (WS) statistics for walkthroughs of different models at 640×480 resolution, 1 ray per pixel, and 2 pixels of error.

6.3. Comparison

The most similar method to ours is that of Yoon *et al.*¹¹, which is also ray tracing based and uses the same LOD error metric as our algorithm. While in their approach the LOD primitives are simple planes called R-LODs, we have instead opted for voxels with varying amount of material attribute sets. This way, we can better approximate complex volumetric details and avoid holes in the simplified versions of the model. Another notable difference is that we load the data asynchronously, thus, our method provides smoother visualization. However, they employ a more efficient *cache-oblivious* data layout, which can lead to faster rendering and loading. The two approaches could be combined to obtain an even more powerful method.

7. Conclusions and future work

We have presented an efficient technique for interactive out-of-core rendering of massive triangular models. Our performance evaluations show that the real-time visualization of different types of massive models is possible on a mainstream notebook PC with a dual-core CPU. We employ a high quality voxel-based LOD mechanism to improve ray casting speed and reduce the amount of required memory. The proposed compressed data structure minimizes the storage and I/O bandwidth requirements. Since data reading and decompression are asynchronous, it is possible to inspect the model even if not all of the necessary details are loaded yet.

In the near future, we would like to extend our approach to also trace secondary rays including shadow, ambient occlusion, reflection, and refraction rays. We also plan to improve rendering quality by implementing anti-aliasing optimized for our voxel-based LOD scheme. Another interesting research avenue is the adaptation of the proposed algorithm for GPUs.

Acknowledgements

The author was supported by the Eurotrans Foundation and the Farkas Gyula Association for Mathematics and Informatics.

References

1. C. Benthin. *Realtime Ray Tracing on Current CPU Architectures*. PhD thesis, Computer Graphics Group, Saarland University, 2006. 3
2. H. Friedrich, I. Wald, J. Günther, G. Marmitt, and P. Slusallek. Interactive Iso-Surface Ray Tracing of Massive Volumetric Data Sets. *Proceedings of the 2007 Eurographics Symposium on Parallel Graphics and Visualization*, pp. 109–116, 2007. 2
3. E. Gobbetti and F. Marton. Far Voxels – A Multiresolution Framework for Interactive Rendering of Huge Complex 3D Models on Commodity Graphics Platforms. *ACM Transactions on Graphics*, **24**(3):878–885, 2005. 2
4. V. Havran. *Heuristic Ray Shooting Algorithms*. PhD thesis, Department of Computer Science and Engineering, Faculty of Electrical Engineering, Czech Technical University in Prague, November 2000. 2, 3
5. A. Reshetov, A. Soupikov, and J. Hurley. Multi-level ray tracing algorithm. *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, pp. 1176–1185, 2005. 3
6. S. Rusinkiewicz and M. Levoy. Streaming QSplat: A Viewer for Networked Visualization of Large, Dense Models. *I3D '01: Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pp. 63–68, 2001. 1, 7
7. S. Rusinkiewicz and M. Levoy. QSplat: A Multiresolution Point Rendering System for Large Meshes. *Proceedings of ACM SIGGRAPH 2000*, pp. 343–352, 2000. 1
8. I. Wald. *Realtime Ray Tracing and Interactive Global Illumination*. PhD thesis, Computer Graphics Group, Saarland University, 2004. 3, 4, 5, 6
9. I. Wald, A. Dietrich, P. Slusallek. An Interactive Out-of-Core Rendering Framework for Visualizing Massively Complex Models. *Proceedings of the Eurographics Symposium on Rendering*, 2004. 1
10. S.E. Yoon, E. Gobbetti, D. Kasik, and D. Manocha. *Real-Time Massive Model Rendering*. Synthesis Lectures on Computer Graphics and Animation. Morgan & Claypool Publishers, 2008. 1
11. S.E. Yoon, C. Lauterbach, and D. Manocha. R-LODs: Fast LOD-based Ray Tracing of Massive Models. *The Visual Computer: International Journal of Computer Graphics*, **22**(9):772–784, 2006. 2, 5, 6, 8
12. S.E. Yoon, B. Salomon, R. Gayle, and D. Manocha. Quick-VDR: Interactive View-Dependent Rendering of Massive Models. *IEEE Visualization*, pp. 131–138, 2004. 1

Real-Time Volumetric Caustics with Projected Light Beams

Gábor Liktó, Carsten Dachsbacher

VISUS, University of Stuttgart

Abstract

We present a method for the visualization of volumetric caustics in single-scattering participating media. The caustics beams are generated from a projected grid in light's image space, making the solution independent from the geometric complexity of generators. The caustic volumes are extruded in the geometry shader and accumulated into the final volumetric effect.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Raytracing I.3.7 [Computer Graphics]: Color, shading, shadowing and texture

1. Introduction

Specular surfaces tend to change the distribution of reflected or refracted photons causing beautiful light phenomena in their environment. Caustics are high-frequency patterns appearing on the surface of diffuse objects as the curvature of the specular surface focuses or defocuses the light. Rendering caustics adds a lot of realism to the final appearance of the image. A glass of drink casts typical focused light to the table, and it is hard to make an underwater scene believable without the wavy light-patterns on the ground. In this scenario the specular objects are generally called caustics generators and the diffuse ones receivers. Being an important global illumination effect, many efforts have been made in recent computer graphics research for plausible real-time approximation.

While recent work mainly focused on the rendering of surface caustics, the efficient handling of volumetric phenomena requires slightly different considerations. If the specular surface is enclosed by participating media, the rays of focused photons become visible to the viewer. The effect can be experienced under the sea surface, since the sea water always contains microscopic particles attenuating the light. In the case of surface caustics, the main challenge is the adaptive sampling and filtering of the projected pattern to make it smooth and continuous on the screen^{8,9}. Volumetric caustics are less sensitive to the accuracy of filtering, but require the integration of the in-scattered light along the viewing ray for each pixel.

A common way of rendering volumetric caustics is to subdivide the caustics generator to small patches and extrude them along the directions of the specular photons, forming the bounding volumes of the caustic beams. One possible subdivision is the usage of the mesh geometry itself. By accumulating the contributions of the beams in each pixel, the volumetric photon distribution becomes visible. The appropriate rendering of such volumes was deeply analyzed in the work of Ernst et al.¹, which we used as main basis for our implementation.

Our contributions include the GPU-based implementation of the volumetric caustics algorithm and the application of the projected grid concept to the beam generation. Instead of using the mesh geometry, our method projects a vertex grid to the surface in light's image space and then performs the extrusion of the beams in the geometry shader. This way the complexity of the algorithm becomes independent of the complexity of the specular surface models.

The paper is organized as follows. The next section summarizes recent work on interactive rendering of caustics, and in particular volumetric effects. Section 3 describes our algorithm in detail. Section 4 highlights some important aspects of the implementation and section 5 presents the test results, followed by a discussion.

2. Previous Work

Most of the caustics rendering algorithms can be divided into two main stages. The first phase starts from the light source,

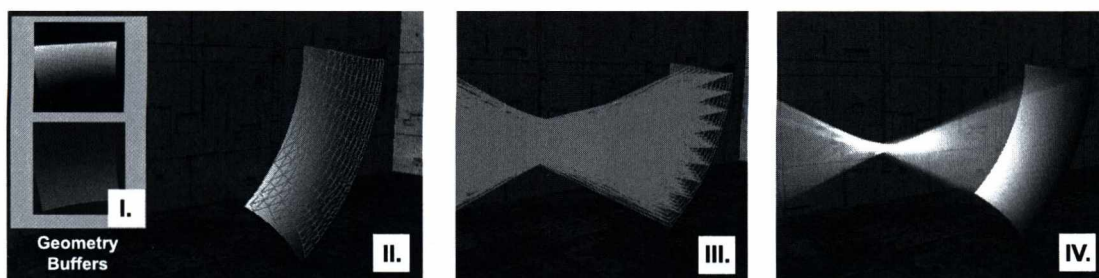


Figure 1: The outline of the algorithm. In the first pass, the specular surface normals and positions are stored in the geometry buffers (I). A regular grid is projected onto the surface (II), from which the geometry shader extrudes the beam volumes (III). The volumetric caustic effect is reconstructed by ray-marching in the pixel-shader (IV).

and identifies the terminal points of caustic light paths on the non-shiny surfaces. In the following step, the contributions of these paths are accumulated at the points visible from the camera.

Photorealistic off-line rendering algorithms generally rely on ray-tracing methods, producing extremely accurate caustics for the price of high computational costs. Photon mapping² is a popular approach, which stores caustics photon hits in dedicated photon maps. Later, the enhanced algorithm was able to capture volume caustics³, and Purcell et al.⁴ implemented it on the graphics hardware.

By the advent of GPU-based solutions, the interactive rendering of caustics became possible. Caustics mapping can be regarded as a simplification of the photon mapping algorithm. Since the k-neighbor gather operation does not suit to the current GPU architectures, the photon hits are stored in a 2D buffer. Possibilities of representing photon locations include texture space⁵, image space^{7, 9}, or the coordinate system of the caustics generators⁶.

During the caustics reconstruction, most of the recent works applied photon-splatting. The common problem of splatting algorithms is to find the optimal photon distribution (importance sampling) and splatting size to achieve high quality results. Wyman and Dachsbacher⁷ improved the quality by the adaptive variation of splat sizes. Recent work of Wyman et al. analyzed the problem of hierarchical sampling during the caustic map generation^{8, 9}.

Other researchers proposed alternative reconstruction methods. Rendering continuous geometry instead of discrete photon hits can eliminate the discretization artifacts. Ernst et al.¹ applied beam tracing¹⁰ to render interpolated caustic triangles to the receiver surfaces. Their algorithm significantly improved the quality of caustic triangle methods by taking into account the warped distortions of the beams. However, it does not handle occlusions, and requires CPU assistance due to the limitations of graphics hardware at that time. In a recent work, Umenhoffer et al.⁶ render caustic triangles on

the GPU using layered distance impostors⁵ to render smooth caustics with occlusion information.

For volumetric caustics effects, beam tracing was used by Nishitha and Nakamae¹¹ to render underwater volume caustics. Iwasaki et al.¹² implemented the former algorithm on the GPU. For a moderate amount of caustic triangles, their results had blocky artifacts because of using constant shading. Ernst et al.¹ analyzed the radiance distribution along a warped triangular beam, and were able to decrease the resolution of the generator mesh by using interpolation inside the beams. The caustic interpolation was implemented in the pixel shader during the rasterization of the convex bounding prisms of the warped volumes.

Approximate global illumination algorithms often try to avoid expensive computations at invisible points of the scene. Screen space adaptive subdivision is one of the possible optimization alternatives. In an early application for the rendering of ocean surfaces¹³ a regular screen space grid was projected to the heightmap of the water. Recently Müller et al.¹⁴ used this concept to render arbitrary 3D fluids with screen space tessellation. The advantage of the method is that surfaces closer to the viewer get finer geometric details, while invisible surfaces are not tessellated at all. We have applied this idea to the problem of beam tracing in this work.

3. Algorithm

Our method for rendering volume caustics breaks down to two rendering passes (Figure 1). In the first pass the surfaces of the specular objects are rendered from the light source. The photon bounces of the surfaces are stored in a set of render targets. Using these as textures, the second pass generates the geometry of the beams. For each beam the scattered radiance towards the camera needs to be determined using the participating media rendering equation. The final caustics effect is the result of the accumulated rendering of each beam.

Since we do not have control over the rendering order of caustic beams, only single-scattering homogeneous participating media can be simulated. For real-time applications, this simplified model is generally satisfactory.

3.1. Beam Generation

The first step of the algorithm captures the caustics generator surfaces from the light source. Since the beam generation is independent of the mesh topology, surfaces of arbitrary number and complexity can be used as generators in the same pass. The surface positions, normals and material properties are stored in geometry buffers. Usually the choice of a proper coordinate system for these parameters is important for the following steps. As we will see later, most of the computations are performed in a special local space for each individual beam, therefore any initial representation is sufficient. Currently we store the world space positions and normals, but in a caustic shadows solution we can transform those into cube-map space like Umenhoffer et al.⁶.

The next step projects a regular grid in light's image space to the specular surfaces. Having the previously rendered geometry buffers, this process is extremely simple. A pre-generated regular grid is sent to the pipeline, but during rendering the position and normal vertex attributes are replaced by the sampled values from the geometry buffers. Prior to rendering the buffers are cleaned to values indicating that the given pixel is invalid. This will allow the geometry shader to discard the primitives not covered by the generator surfaces.

The projected grid is then forwarded to the geometry processing stage. The geometry shader either discards the triangles which are not bases of caustics beams, or emits new primitives forming the convex boundaries of the caustic volumes.

If the vertices of the specular triangle are denoted by $\vec{v}_0, \vec{v}_1, \vec{v}_2$ and the caustic photon directions at these vertices $\vec{d}_0, \vec{d}_1, \vec{d}_2$, the geometry shader has to extrude a volume bounding the rays $\vec{v}_i + t_i \vec{d}_i, i \in \{0..2\}$. In our current solution t_i is a constant value, since we do not consider occlusions yet. Later on t_i should be determined by a depth-texture lookup.

The extrusion of the beams needs special considerations¹. In the general case the geometry of the beams cannot be represented by a finite set of planar surfaces, since the adjacent \vec{d}_i rays do not necessarily lie on the same plane, but form the sides of a bilinear patch instead. Figure 2 illustrates artifacts raised by the direct emission of the beams in the geometry shader. From the point of volume caustics, the main artifact is caused by the additive blending, where the same beam might render multiple times to the same pixel, resulting in light streaks. This artifact remains dominant on the screen even at high grid resolutions.

The task of the geometry shader is to emit a triangular mesh bounding the bilinear patches. During the extrusion a

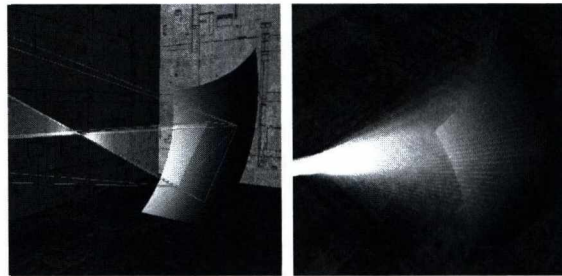


Figure 2: The rendering artifacts when extruding along the photon directions. The non-planar surfaces of the bilinear patches are approximated with two triangles (left), which leads to light streaks when using additive blending (right). The contrast of the image was slightly increased to emphasize the effect.

touching plane is found for each edge of the specular triangle. Assuming finite beam length, one can always find a plane that holds the entire caustics volume on its negative half space¹. The extrusion ray for each vertex can be found as the intersection of touching planes for the adjacent edges.

The volumetric rendering equation is solved in the fragment shader, therefore the geometry processing stage must encode all the necessary parameters into the emitted vertex attributes for performing the sampling of the beams. These parameters are:

- The positions of the caustic vertices
- The directions of the caustic beam edges
- The area of the specular triangle
- The caustic radiance values at the vertices of the triangle

The main task of the fragment shader is to find the radiance at each sample point in the beam. This sampled radiance depends on the initial caustic radiance on the surface and the area ratio of the specular triangle and the parallel cross section of the beam at the sample point. We know that the radiant flux

$$\Phi(\Delta\omega, \Delta_c) = \int_{\Delta_c} \int_{\Delta\omega} L(\vec{p}, \vec{\omega}) \cos\theta d\vec{\omega} dA$$

is constant for every cross section of the beam (Δ_c is the caustic triangle, \vec{p} is a representative point in the infinitesimal surface and θ is the angle between the surface normal and $\vec{\omega}$). If the cross section is parallel to the triangle, the radiance at each point can be computed using the area ratio. The question is how to get the area of the cross section of the warped volume?

The double area of the caustic triangle is defined by the length of the cross product of the edges:

$$A(\Delta_c) = |(\vec{v}_1 - \vec{v}_0) \times (\vec{v}_2 - \vec{v}_0)|$$

The calculation of the double area for each sample along the beam would be very expensive in the fragment shader. It is desirable to express the area as a function of distance along the beam. Ernst et al.¹ proposed a special coordinate system, which we will call beam space in the following, where the mentioned cross product simplifies to a one dimensional problem.

Since we are looking for cross sections parallel to the specular triangle, it is beneficial to use a basis where one axis is perpendicular to the triangle. Prior to further calculations, the beam parameters are transformed into a coordinate system where \vec{v}_0 is the origin, the x -axis is the $\vec{v}_0 - \vec{v}_1$ edge and the y -axis is the triangle normal. Normalizing these vectors and choosing their cross product as z -axis gives us an orthogonal basis in beam space. Another important simplification is to scale the \vec{d}_i ray direction vectors so that their y -coordinate equals to one in beam space (Figure 3).

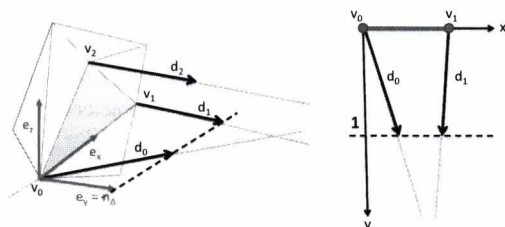


Figure 3: The basis vectors in beam space. The ray-marching of the beams is performed in this coordinate system.

Using this representation, we can rewrite the area function into the following form¹:

$$\begin{aligned} A(y) &= |(\vec{v}_1 + y\vec{d}_1' - \vec{v}_0 - y\vec{d}_0') \times (\vec{v}_2 + y\vec{d}_2' - \vec{v}_0 - y\vec{d}_0')| \\ &= |(\vec{k}_1 + y\vec{l}_1) \times (\vec{k}_2 + y\vec{l}_2)|, \\ &\quad \text{where } \vec{k}_i = \vec{v}_i' - \vec{v}_0' \text{ and } \vec{l}_i = \vec{d}_i' - \vec{d}_0' \end{aligned}$$

Vector \vec{v}' is \vec{v} in beam space using the new basis. Since $\vec{k}_{iy} = \vec{l}_{iy} = 0 \forall i$, the length of the cross product will equal to its y -component.

Expanding the formula we get a second order expression over y in the form of $ay^2 + by + c$, where:

$$\begin{aligned} a &= \vec{l}_{1z}\vec{l}_{1x} - \vec{l}_{1x}\vec{l}_{2z}, \\ b &= \vec{k}_{1z}\vec{l}_{2x} - \vec{k}_{1x}\vec{l}_{2z} + \vec{l}_{1z}\vec{k}_{2x} - \vec{l}_{1x}\vec{k}_{2z}, \\ c &= \vec{k}_{1z}\vec{k}_{2x} - \vec{k}_{1x}\vec{k}_{2z} \end{aligned}$$

The a , b , c area coefficients are computed in the geometry shader and stored in a vertex attribute. The new coordinate system must also be forwarded to the fragment shader, which evaluates the volumetric rendering equation by ray-marching in beam space.

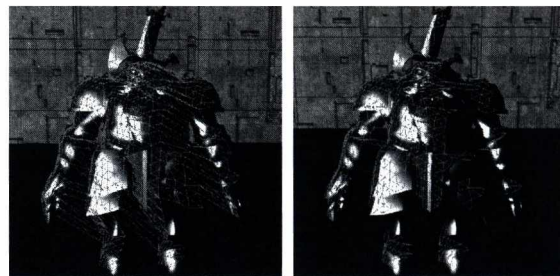


Figure 4: The direct projection of the regular grid results in false triangles around depth discontinuities. Using simple heuristics based on the triangle normals and the light's direction vectors, these triangles are eliminated.

The geometry shader must also remove triangles which would result in incorrect caustics, for example by not lying on the same surface. The projected grid is independent of the scene, therefore it is possible that some triangles go through extreme distortions. We use the dot product of the triangle normal \vec{n}_Δ and the light direction vector \vec{d}_l as heuristics for discarding "wrong" triangles. We assume that if \vec{n}_Δ is almost parallel to \vec{d}_l then the caustics effect is the most accurate, while if they are "almost" perpendicular to each other, the triangle must be discarded. In figure 4 we visualized the projected grid using the normal heuristics.

3.2. Ray-Marching Caustics Volumes

The fragment shader must first intersect the camera-ray with the beams, then solve the volumetric rendering equation in a set of sample points inside them. For homogeneous participating media this equation takes the following form:

$$L(\vec{\omega}) = \int_{\Delta x} e^{-\sigma_t(s_1+s_2)} \sigma_s P(\vec{\omega} \cdot \vec{\omega}') L_{in}(\vec{\omega}') ds$$

where $L(\vec{\omega})$ is the radiance seen from the camera, Δx is the interval of the ray-beam intersections, σ_t is the extinction coefficient, σ_s is the scattering coefficient, P is the phase function, and $s_1 + s_2$ is the travel length of caustic photons from the specular surface to the camera. The notations are visible in Figure 5. Ray-marching is a numerical approximation of the above integral. In the case of volumetric caustics, we have found that it is enough to take a single sample in the middle of the Δx interval. The simplified numeric approximation becomes

$$L(\vec{\omega}) \approx \Delta x e^{-\sigma_t(s_1+s_2)} \sigma_s P(\vec{\omega} \cdot \vec{\omega}') L_{in}(\vec{\omega}')$$

The sampling in the fragment shader is performed in three steps (The whole process is illustrated in figure 5):

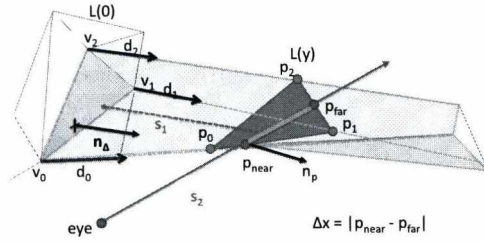


Figure 5: Intersecting the caustic volume with a ray. The intersection of the beam edges and the ray-plane defines a triangle. Using 2D line-line intersections, the distance which the ray takes inside the volume is determined. The participating media equation is evaluated at a sample point inside this Δx -long interval. To get the caustic radiance values, the area function is evaluated at the corners of the intersection triangle.

- Transform the viewing ray into beam space.

The basis of the new coordinate system is given as a vertex attribute.

- Intersect the beam with a plane containing the viewing ray.

The normal of such a plane is determined by:

$$\vec{n}_p = r_{view} \times (r_{view} \times \vec{n}_\Delta)$$

where r_{view} is the viewing vector and \vec{n}_Δ is the triangle normal (the y basis in beam space). By intersecting the \vec{d}_{0-2} caustic photon directions with this plane, we get a triangle in the same plane as the viewing ray.

- Intersect the resulting triangle with the ray and trilinearly interpolate the radiance at sample point \vec{p} .

The radiance values at the corners of the intersection-triangle are determined efficiently by evaluating the area function. Since the area coefficients were precalculated in the geometry shader, the fragment shader only has to evaluate the dot product $(y^2, y, 1) \cdot (a, b, c)^T$.

Since the triangle and the ray lie in the same plane, the 2D intersection simplifies to finding the intersections with the edges (2D line-line intersection), and interpolating between the caustic radiances at the triangle corners. If the ray does not intersect the triangle, the fragment is discarded.

4. Implementation

We implemented the algorithm in DirectX 10 using HLSL shaders, with our test platform equipped with Nvidia Geforce GTX 260 graphics card, Intel Core i7 920 CPU and 6 GB RAM.

The geometry buffer generation is a straightforward process. Note that instead of storing the data in textures, the geometry could have been rendered directly into the grid vertex buffer. This would avoid the execution of the vertex shader program in the second pass with multiple texture lookups. However, we did not use this solution for flexibility reasons: in the future we plan to implement adaptive hierarchical beam generation in the geometry shader which will make texture lookups necessary anyway.

The second rendering pass which performs the actual volume caustics rendering must be executed once for each caustic effect. For example if we want to visualize both reflective and refractive caustics the projected grid is to be rendered two times. Two sided refractions are not supported yet. The output format of the beam geometry shader is unusually complex, since it must encode the entire beam into one vertex – the fragment shader gets these parameters as interpolated attributes. The beam parameters are packed into 10 *float4* vertex attributes as shown in table 1.

Field Name	Components			
Params0	$v1^b.x$	$v1^b.z$	$v2^b.x$	$v2^b.z$
Params1	$e0^b.x$	$e0^b.z$	$e1^b.x$	$e1^b.z$
Params2	$e2^b.x$	$e2^b.z$	2A	
L0	Radiance of edge 0			
L1	Radiance of edge 1			
L2	Radiance of edge 2			
Basis0	ex		v0.x	
Basis1	ey		v0.y	
Basis2	ez		v0.z	
wpos	World position			
AreaCoeffs	a	b	c	

Table 1: Vertex attributes.

In the *Params0-2* attributes the beam geometry is stored in beam space. Note that we do not need to store the coordinates of \vec{v}_0 and any y -components since these are zero in this basis. *Params2* also stores the double area of the caustic triangle. *L0-2* holds the caustic radiances at the corners of the specular triangle. *Basis0-2* are also very important, since they encode the transformation into beam space. Finally *wpos* is the world position of the vertices (the viewing ray is the difference between the world position and the eye position) and *AreaCoeffs* holds the area coefficients.

5. Results

The test renderings used for performance measurement are shown in figure 6. The results are summarized in table 2. Note that the test scenes contain other shaders not related

to our method, like a naive implementation of volumetric shadows with ray-marching.

Grid res.	Total	Caustics	Without PS	
128x128	92.5	80	11.3	(10.8 FPS)
64x64	45	31.4	9.1	(22.2 FPS)
32x32	24.1	9.7	6.2	(41.4 FPS)

Table 2: Frame times of the paraboloid scene (figure 6) with different grid resolutions. The first three columns are in milliseconds. Besides the total frame time, we measured separately the time of rendering the caustic beams (Caustics) and the time without the caustics pixel shader (Without PS). Note, that the high fill-rate is the bottleneck of the algorithm.

To measure the performance of the geometry shader – which is definitely a critical stage on Shader Model 4 GPUs – we also run the tests without the costly ray-marching pixel shader. These results can be read under the caption "without PS".

6. Discussion and Future Work

Using the projected grid concept the performance is not significantly influenced by the geometric complexity. The main problem of rendering volume caustics is finding the golden mean between shading quality and fill-rate. When using a large grid resolution, the caustic effect becomes accurate, but several beams are rasterized to the same pixel. On the other hand, reducing the grid resolution introduces sampling artifacts which are particularly noticeable at geometric discontinuities (silhouettes).

As a future work, we would like to add shadowing support and multiple specular bounces to our implementation. An important form of the latter is the support of double-sided refractions. During the geometry shader execution, approximate ray-tracing methods can be used to determine the termination points for the caustic beams.

The other main future direction is the implementation of the adaptive hierarchical sampling in light's image space, similarly to Wyman's work⁹. Initially using a coarse vertex grid, the geometry shader can look for discontinuities in the geometry buffer to refine the resolution of the projected grid. Since we are using the grid for beam extrusion and not for direct rendering, the cracks introduced by such subdivisions do not cause artifacts. An adaptive algorithm would also reduce the overhead of discarded triangles in the geometry shader.

The next important way to attack the fill-rate bottleneck is off-screen rendering. Instead of rendering caustics to the screen directly, the beams can be rendered into a downsampled buffer, significantly improving the performance, while only slightly degrading the quality.

Acknowledgement

This work has been supported by Crytek GmbH.

References

1. M. Ernst, T. Akenine-Möller and H. W. Jensen Interactive Rendering of Caustics using Interpolated Warped Volumes *Proceedings of Graphics Interface 2005*, pp. 87-96 (2005) 1, 2, 3, 4
2. H. W. Jensen Global Illumination Using Photon Maps *Proceedings of 7th Eurographics Workshop on Rendering*, pp. 21-30 (1996) 2
3. H. W. Jensen, P. H. Christensen Efficient Simulation of Light Transport in Scenes with Participating Media using Photon Maps *Computer Graphics (SIGGRAPH 98)*, pp. 311-320 (ACM Press, 1998) 2
4. T. J. Purcell, C. Donner, M. Camarano, H. W. Jensen and P. Hanrahan Photon Mapping on Programmable Graphics Hardware *Graphics Hardware*, pp. 41-50 (Eurographics Association, 2003) 2
5. L. Szirmay-Kalos, B. Aszódi, I. Lazányi and M. Premecz Approximate Ray-Tracing on the GPU with Distance Impostors *Computer Graphics Forum (Eurographics 2005)*, pp. 695-704 (2005) 2
6. T. Umenhoffer, G. Patow and L. Szirmay-Kalos Caustic Triangles on the GPU *Proceedings of of Computer Graphics International*, (2008) 2, 3
7. C. Wyman, C. Dachsbacher Improving Image-Space Caustics via Variable-Sized Splatting *Technical Report UICS-06-02*, University of Utah (2006) 2
8. C. Wyman Hierarchical Caustic Maps *Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games*, pp 163-171 (2008) 1, 2
9. C. Wyman, G. Nichols Adaptive Caustic Maps Using Deferred Shading *Computer Graphics Forum* 28(2), pp. 309-318 (2009) 1, 2, 6
10. P. S. Heckbert, P. Hanrahan Beam Tracing Polygonal Objects *Computer Graphics (SIGGRAPH 84)*, pp. 119-127 (ACM Press, 1984) 2
11. T. Nishita, E. Nakamae Method of Displaying Optical Effects within Water using Accumulation Buffer *Computer Graphics (SIGGRAPH 94)*, pp. 373-379 (ACM Press, 1994) 2
12. K. Iwasaki, Y. Dobashi and T. Nishita An Efficient Method for Rendering Underwater Optical Effects Using Graphics Hardware *Computer Graphics Forum*, 21(4), pp. 701-712 (2002) 2
13. C. Johanson Real-Time Water Rendering - Introducing the Projected Grid Concept *Master of Science Thesis (Lund University)*, (2004) 2

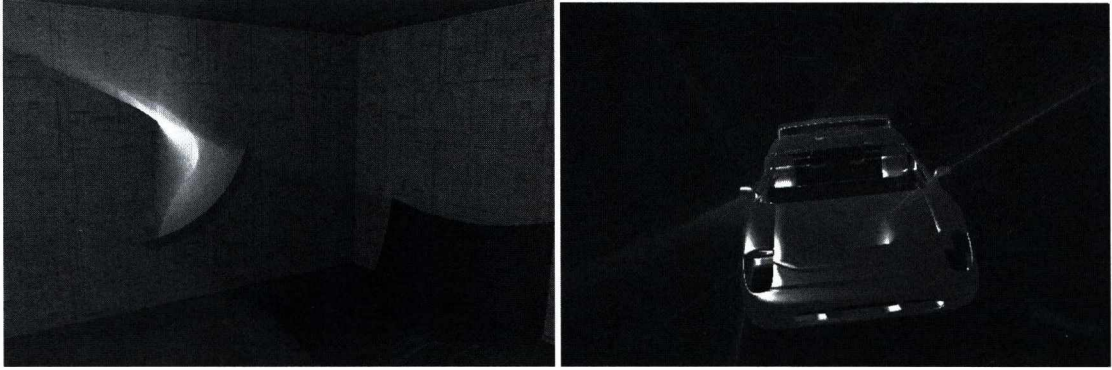


Figure 6: Test renderings of various volume caustics using our algorithm. Left: a paraboloid surface focuses the light. (Rendering time with volumetric shadows: 30 FPS). Right: focused reflections of a car body. For this quality a 128×128 -sized projected grid was necessary. Because of the extremely high fill-rate the rendering time dropped to 8 FPS. Both images were rendered in 900×600 .

14. M. Müller, S. Schirm and S. Duthaler Screen Space Meshes *Proc. of the 2007 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp. 9-15 (2007) 2

Recursive Ray Tracing in Geometry Shader

Kristóf Ralovich[†] and Milán Magdics[‡]

Department of Control Engineering and Information Technology
Budapest University of Technology and Economics
Budapest, Hungary

Abstract

We propose an effective method to enable recursive ray tracing triangular scenes using contemporary real-time graphics pipelines of digital computers. So far, general purpose computations such as ray tracing utilized the programmable pixel shader for computation. Lengthy algorithms were subdivided to continue-and-restart-able parts (computing kernels) to enable implementation as multi-pass rendering. Discussing a fundamentally different method, we are representing rays with geometry, maintaining a one-to-one correspondence between rays and point primitives. Exploiting the geometry amplification capability of modern pipeline, we are utilizing the geometry shader to emit multiple secondary rays. Our approach to recursive ray tracing is influenced by stream computing, circulating the data flow without using a stack, employing intermediate pipeline stage to feedback transformed primitives before producing the final color by rasterization of point primitives. An effective implementation employing the uniform grid space subdivision scheme is described.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism: Raytracing

1. Introduction

Global illumination techniques and thus recursive ray tracing is gaining increasing popularity in real-time image synthesis practice due to widespread availability of consumer level fast parallel digital computer and the algorithmic improvements of the recent years.

Ray shooting maps well and easily to GPUs by the independent nature of rays, and using the graphics pipeline to implement ray tracing had a well known stream computing approach so far: in each rendering pass rasterizing a view-port sized quadrilateral to activate the fragment shader on a regular 2D matrix of pixels where every pass operates on one single ray generation only. A ray generation is for example the first level of shadow rays only (without mixing different

type of rays). Also, e.g. the first bounce of specular reflected rays form an other ray generation.

This paper discusses a fundamentally different configuration of the pipeline, where individual rays are represented by point primitives as opposed to pixels. Recursion required for¹² ray tracing is achieved by feeding back the transformed primitives to the beginning of the pipeline instead of using a large number of textures (render targets) to emulate a stack to store the state of computation. Decomposing⁵ the ray tracing algorithm was historically required because of tight restrictions posed by the GPU on the static and dynamic instruction counts in a shader program. These constraints are not a problem any more, but underlying architectures are still poorly suited for stack memory and thus shading languages can not support recursive function calls directly.

[†] kristof.ralovich@gmail.com

[‡] gumi@inf.elte.hu

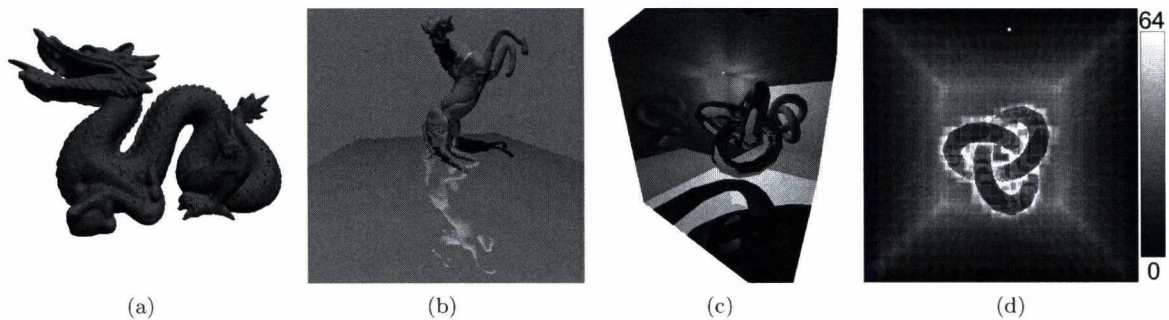


Figure 1: (a)(b)(c) Images generated at interactive rates using the discussed algorithm using geometry shader and transform feedback. Rays are represented in the pipeline with geometry, viewport is 512^2 sized. Images from left to right in respective order: the "Dragon" shadowcasted, the "Horse" scene with specular materials, and the "Torus Knot" scene with reflections and shadows from a single light source. (d) Depicts the cost of finding the intersection for each ray, speed up impact of the employed uniform grid space subdivision is apparent.

2. Previous work

Applying the processing power of programmable GPU to speed up ray tracing motivated many researchers of the field of real time computer graphics.

Carr et. al.¹ discovered that ray-object intersections may be computed in a fragment shader, later others managed to apply a great variety of different techniques such as space subdivision and partitioning (uniform 3D grid^{5,6}, k D-tree^{3,9}, BVH¹¹, etc.) to the GPU to reduce the number of intersections finding the closest visible hit. Szécsi⁸ and Roger et. al.⁷ experimented with ray space hierarchies for logarithmic speedup culling away rays, while Carr et. al.² used geometry images to store the scene GPU memory friendly. The common concept of these methods was using the fragment shader to do the necessary computations.

Szirmay et. al.¹⁰ provides a comprehensive overview of image space methods, practical global illumination, and ray tracing on the GPU.

Recent research shows high performance ray tracing implementations⁴ are possible using the CUDA general purpose parallel programming architecture. CUDA provides direct C language interface to the graphics hardware without special knowledge of programmable graphics pipeline, but we are still focusing on a method layered over graphics APIs because of their wider general availability.

3. Algorithm overview

Let us summarize the structure of our algorithm:

1. Set up two vertex buffers (VB) with enough space.
2. Fill one of the VBs partially with point primitives representing primary rays and bind it as drawing source.

```

struct Hit
{
    vec4 pos; // point primitive
    vec3 orig; // ray origin
    vec3 dir; // ray direction
    vec2 uv; // barycentric hit coords
    float t; // ray parameter
    int idx; // hit triangle index
    int state; // state (flags)
    int type; // hit primitive type
};
    
```

Listing 1: Data structure holding the hit record.

```

struct PackedHit
{
    vec4 pos;
    vec4 orig_t;
    vec4 dir_idx;
    vec4 uv_state;
};
    
```

Listing 2: Structure encapsulating a ray and corresponding hit record. Encoded as four component floating point vectors. Passed to the pipeline as a point primitive with associated vertex attributes

3. Bind the other VB for stream output destination.
4. Set up the pipeline with the discussed shaders (section 4.1).
5. Rasterize (draw) the point primitives in P number of passes without changing the shaders and without any CPU intervention. A rendering pass consists of the following tasks:
 - a. Calculate intersection in the vertex shader.

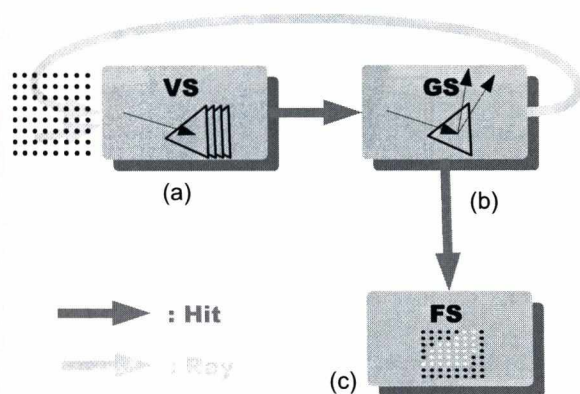


Figure 2: Recursion in the pipeline. Rays are represented by point primitives with additional vertex attributes to hold the hit record. (a) Intersection in the vertex shader. (b) Geometry shader emitting secondary rays and terminating finished rays. (c) Final shading and color compositing in fragment shader.

- b. Depending on material properties, emit secondary rays in the geometry shader.
 - c. Use the stream out stage to feedback transformed points primitives.
 - d. Fragment shader computes color at valid hits.
 - e. Blend the resulting pixel colors together.
6. Swap the VBs. This is called ping-ponging.

4. Algorithm details

The reason we are required to use two vertex buffers is that the source of drawing and the destination of stream output cannot be the same object.

Prior to allocating the VBs, we must know (Figure 4(a)) their maximum sizes: $S_{VB} = w \times h \times (G + G') \times S_V$, where w and h are the dimensions of the viewport, G and G' are the maximum and second largest number of ray generations emitted in each depth of ray tracing recursion, and S_V is the size of a vertex (i.e. the ray structure including a hit record, which is 64 bytes in our case, see code Listing 2). Note that this calculation is not dependent on the number of triangles, and scales with the screen size.

4.1. Pipeline setup

The algorithm is implemented using three stages of the programmable pipeline (Figure 2) and the transform feedback (stream out) functionality. Listings 3, 4, and 5 describe the three shaders in pseudo code.

One could have implemented the functionality of the vertex shader in the geometry shader as well, however

our separation of functionality is geared towards high performance: shorter shaders result in lower branching divergence behavior and thus more coherent memory access patterns.

4.2. Shaders

In the **vertex shader** vertex attributes of point primitives (rays) are loaded from the VB. Ray intersection with the scene is carried out, hit record is updated accordingly and output. As an additional optimization, the ray is checked for intersection against axis aligned bounding box of the scene. If there is no hit, the ray is marked for termination in the subsequent geometry shader.

The output of the vertex processor is considered as the hit record. Depending on the ray state and material properties of the intersection, the ray is terminated (not emitted) or further secondary rays (shadow, reflection) are emitted. Emitted new rays are coupled with a hit record indicating intersections should be calculated in the next vertex shader pass, and to be skipped in the following pixel shader in this pass (because shading requires a valid hit record first). Rays that have been written to the framebuffer in the previous pass are not processed in the **geometry shader** and are terminated (not emitted) early without processing. Transform feedback (Stream Output) is configured to allow receiving multiple output primitives per each input. If there is anything to output, the geometry shader is emitting rays in a fixed local order: first the input ray, later the shadow and finally the specular reflected ray.

Color of rays with valid hit records is calculated employing the Phong shading model and written into the framebuffer in the **pixel shader**, then a special blending operation composites visible color, shadowedness, and the secondary color.

4.3. Rendering

In order to visualize rays in the framebuffer, a viewport sized 2D grid of point primitives is rasterized on a plane perpendicular to the view direction of the virtual camera. The positions of the point primitives are set up so that rasterization assigns them to the pixel centers, which will result in the whole coverage of the screen.

We are exploiting that the number of primitives output from the geometry shader is not needed to be queried back to the CPU to issue a draw call sourcing those vertices.

As stated in section 4.1, the geometry shader has a fixed output order of rays. The graphics pipeline has a

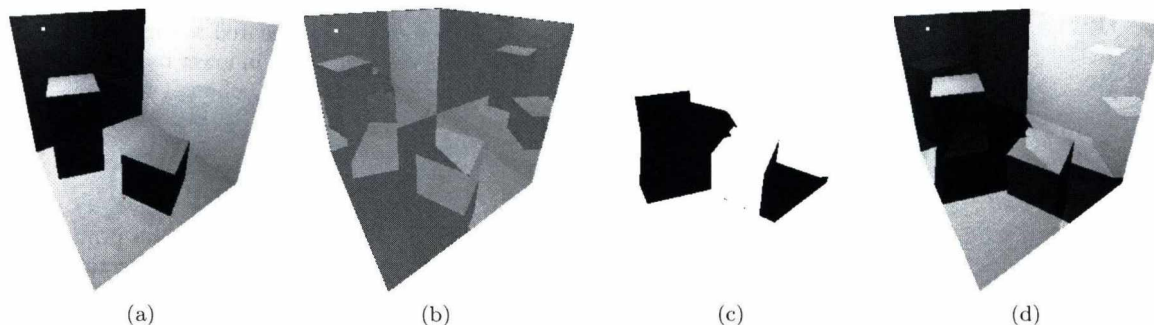


Figure 3: The contribution of eye rays (a), reflected rays (b) and shadow rays (c). The right most (d) image shows the final composited image.

very unique property[†]: primitives hit the framebuffer in the order listed in the VB (if no other indexing is used, like in our case), this is necessary for our algorithm to work, since the processing order of rays must not be changed. Imagine the specular contribution is added to a pixel before processing the previous shadow ray: the shadow ray would amortize the influence of the reflected ray instead of the previous hit.

The P number of rendering passes – required for the correct image – equals to the depth of the ray tracing depth. That is e.g. 2 for primary rays and one bounce (since shadow rays and reflected rays spawned by eye hits are handled together in the same pass). See Figure 4(a).

In order to enable the GPU accessing the scene data, texture memory is employed in a read only manner (in the same way as⁶). We are able to use world space coordinates for the scene, so we do not have to transform rays into object space. Using a uniform space subdivision scheme requires storing the list of referenced objects for each voxel of the grid. This is stored in a RGB 3D texture, where each grid cell is encoded in one texel. The list of referenced objects is stored tightly packed and each texel corresponds to a pointer to actual triangle data. After two levels of indirection, actual triangle data is stored in two other 3D textures both featuring 4 depth slices. Texels with the same (X, Y) coordinates in different Z slices are storing vertex, normal and material information belonging to the same single triangle. Thanks to this “co-location” texture coordinates used for addressing a triangle’s attributes need to be computed only once in the shaders.

[†] Such kind of synchronization can easily be costly to implement in e.g. CUDA.

4.4. Blending

Different ray generations are intermixed in the vertex buffer, and thus final color compositing must be capable of both extinguishing radiance (in case of shadows) and adding radiance (for reflections rays). Considering the case of a single point light source, shadows and one bounce of reflections additional to ray casting, Equation 1 shows one possible[‡] simple compositing setup using fast hardware blending (note that RGB_{src} must be pre-multiplied with the ρ reflectivity to get correct results) in only a single rendering pass.

$$RGB_{out} = (1.0 \times RGB_{src}) + (\alpha_{src} \times RGB_{dst}) \quad (1)$$

$$\alpha_{src} = \begin{cases} 1.0 & \text{for eye rays} \\ 0.0 & \text{for shadow rays} \\ 1.0 & \text{for reflected rays} \end{cases}$$

$$RGB_{src} = \begin{cases} (R_{src}, G_{src}, B_{src}) & \text{for eye rays} \\ (1.0, 1.0, 1.0) & \text{for shadow rays} \\ \rho \cdot (R_{src}, G_{src}, B_{src}) & \text{for reflected rays} \end{cases}$$

This equation lets shadow rays to extinguish the contribution from primary rays. The case of more ray generations and/or light sources require different and a more complicated compositing approach. Ray generations should be sorted and rendered to multiple render targets that blending passes may blend together.

5. Results

Comparison of the distribution of time spent[§] in the shaders is summarized in Table 1. From these results

[‡] An other option is the use of `GL_ARB_color_buffer_float` extension, that widens the possibilities of custom blending, and also provides mechanism to disable clamping of color values before blending and use of negative alpha values.

[§] The presented data is the % of executed instructions distributed between shaders, but since the hardware is run-

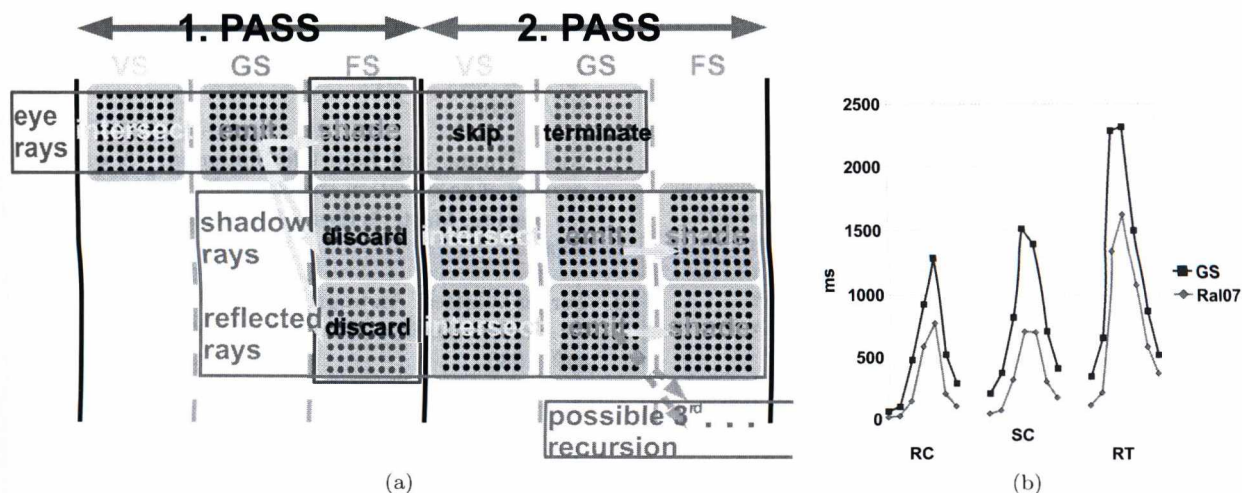


Figure 4: (a) The required number of rendering passes equals to the recursion depth, 2 in the figure (blue, green, and pink frames represent the first, second, and possible third respectively). Each row depicts a different ray generation. Blue areas with dark dots represent the input of stages of a shader program (vertices, fragments) labeled by the task carried out in that shader. Dark red frame shows the maximum size each of the ping-ponging VBs must be capable to accommodate. In the presented case $S_{VB} = 512 \times 512 \times (2 + 1) \times 64$ bytes = 48 Mbytes. (b) Rendering time (in milliseconds) comparison of our method with previous work⁶. The benchmark environment is the same as for Table 3. We suspect that the 2-3 fold worse performance is due to our using of “fat”, (64 byte) primitives.

of naive intersection we conclude, that computation times are limited by vertex shader, that is by the naive intersecting. This is a clear indication that more work may be loaded on the geometry and fragment shaders.

The experiments also showed that the pipeline is compute bound, with this amount of computation the cost of more texture reads may be covered transparently. This leaves us opportunities for improved shading calculations using BRDF sampling.

6. Conclusion and Further Works

4th generation GPUs are built on a unified device architecture. That means the GPU contains only one type of processing unit and that very same unit executes the different types of shaders. This mapping of computations to hardware resources would suggest that it does not matter which type of shader we use to execute the computations and texture reads from, the performance should stay constant. Figure 4(b) shows that this is not true, depicted are our vertex shader[¶] based results compared to the previous results⁶ based

on the fragment shader. We have to conclude that using the vertex shader for intersection calculations is slower than using the pixel shader. This is probably the side effect that we are using “fat” vertices, each point primitive is made up of 64 bytes. We have observed although the geometry shader also has some overhead, the vertex shader even without a geometry shader performs 2-3 times worse^{||} than the fragment shader (see Table 2. for measurements of pure ray casting).

Also moving the light source in the scene incurs high variability in rendering times, this is due that the cost of shadow rays is highly dependent on the position of the light source.

Although efficiency of our setup is apparent, further investigating should be conducted with the current state-of-the-art space subdivision and partitioning methods (kD-tree, BVH, ray hierarchies, etc.) to enable fair comparison with previous techniques using full screen quads.

Extending our work in the future with a best efficiency scene hierarchy would be very interesting.

Our system may be optimized to calculate primary

¶ Shader where the most expensive operation, the intersection test is executed.

|| On a Geforce 8600 GPU.

intersections through rasterization of eye rays for the highest possible performance on the graphics hardware. This would only require changing the first pass of the algorithm.

We found that implementing a ray tracer in a graphics API involves a notable runtime overhead. This limitation can easily be overcome once high performance streaming computing software (OpenCL, CUDA) gains more widespread availability.

References

1. Nathan A. Carr, Jesse D. Hall, and John C. Hart. The ray engine. *Graphics Hardware*, pages 1–10, 2002.
2. Nathan A. Carr, Jared Hoberock, Keenan Crane, and John C. Hart. Fast gpu ray tracing of dynamic meshes using geometry images. *Proc. Graphics Interface*, pages 203–209, 2006.
3. Tim Foley and Jeremy Sugerman. Kd-tree acceleration structures for a gpu raytracer. In *HWWS '05: Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 15–22, New York, NY, USA, 2005. ACM.
4. Johannes Günther, Stefan Popov, Hans-Peter Seidel, and Philipp Slusallek. Realtime ray tracing on GPU with BVH-based packet traversal. In *Proceedings of the IEEE/Eurographics Symposium on Interactive Ray Tracing 2007*, pages 113–118, September 2007.
5. Timothy J. Purcell, Ian Buck, William R. Mark, and Pat Hanrahan. Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics*, 21(3):703–712, July 2002. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002).
6. Kristóf Ralovich. Implementing and analyzing a gpu ray tracer. *Central European Seminar on Computer Graphics*, 2007.
7. David Roger, Ulf Assarsson, and Nicolas Holzschuch. Whitted ray-tracing for dynamic scenes using a ray-space hierarchy on the gpu. In Jan Kautz and Sumanta Pattanaik, editors, *Rendering Techniques 2007*, pages 99–110. the Eurographics Association, jun 2007.
8. László Szécsi. The hierarchical ray engine. *WSCG*, pages 249–256, 2006.
9. László Szécsi and Kristóf Ralovich. Loose kd-trees on the gpu. *IV. Hungarian Conference on Computer Graphics and Geometry*, pages 94–101, 2007.
10. L. Szirmay-Kalos, L. Szécsi, and M. Sbert. *GPU-Based Techniques for Global Illumination Effects*. Morgan and Claypool Publishers, San Rafael, USA, 2008.
11. Niels Thrane and Lars Ole Simonsen. A comparison of acceleration structures for gpu assisted ray tracing. Master's thesis, University of Aarhus, 2005.
12. Turner Whitted. An improved illumination model for shaded display. *Commun. ACM*, 23(6):343–349, 1980.

Appendix A: Shader code

```

#define inVS() \
Ray ray=Ray(orig_t.xyz, dir_idx.xyz);\
Hit hit=Hit(uv_state.x, uv_state.y, \
            orig_t.w,int(dir_idx.w));\
int state=int(uv_state.z); \
int type=int(uv_state.w);

#define outVS() \
gl_Position = gl_Vertex; \
orig_t1.xyz = ray.orig; \
orig_t1.w = hit.t; \
dir_idx1.xyz = ray.dir; \
dir_idx1.w = float(hit.idx); \
uv_state1.xy = vec2(hit.u, hit.v); \
uv_state1.z = float(state); \
uv_state1.w = float(type);

void main()
{
    inVS();
    if(state == 0){
        ray=Ray(cameraPos,normalize(vec3(
gl_Vertex.x,gl_Vertex.y, -1.0)*rot3));
        hit.t = INF;
        hit.idx = -1;
        state = 1;
        type = 0;
        hit = intersect_grid(ray, hit.t);
    }
    #if defined(SHADOWS) || defined(
        RECURSION)
        else if(state == 1) {
            hit = intersect_grid(ray, hit.t);
        }
    #endif
    outVS();
}

```

Listing 3: GLSL code for the vertex shader.

```

void main()
{
    inGS();

    if(state > 1)
        return;

    if(hit.idx == -1)
        return;

    emitPassThrough();

    if(type != 0 || emitNoMore>0)
        return;

    #if defined(SHADOWS) || defined(
    RECURSION)
        lookupNormal(hit, hitN);
        vec3 hitP = ray.orig + ray.dir*hit.t
            + hitN*EPSILON;
    #endif
    #ifndef SHADOWS
        vec3 toLight = lightPos - hitP;
        float lightDist = length(toLight);
        Ray shadowRay = Ray(hitP, toLight/
            lightDist);
        Hit shadowHit = Hit(0.0, 0.0,
            lightDist, -1);
        state = 1;
        type = 1;

        emitShadowRay();
    #endif
    #ifndef RECURSION
        Ray reflRay = Ray(hitP, reflect(ray.
            dir, hitN));
        Hit reflHit =Hit(0.0, 0.0, INF, -1);
        state = 1; // intersect in
        // real pos. t3 instead in the
        // pos.
        type = -1;

        emitReflRay();
    #endif
}

```

Listing 4: GLSL code for the geometry shader.

```

void main()
{
    Ray ray=Ray(orig_t2.xyz,dir_idx2.xyz);
    Hit hit=Hit(uv_state2.x,uv_state2.y,
        orig_t2.w,int(dir_idx2.w));
    int state = int(uv_state2.z);
    int type = int(uv_state2.w);

```

```

if(state < 3)
    discard;
if(type == 0)
{
    Ray eyeRay = ray;
    Hit eyeHit = hit;
    if(eyeHit.idx == -1)
    {
        gl_FragColor = vec4(backgroundColor.
            rgb, 0.0);
        return;
    }
    vec3 eyeHitPosition = eyeRay.orig +
        eyeRay.dir * eyeHit.t;
    vec3 lightVec = lightPos -
        eyeHitPosition;
    lookupNormal(eyeHit, N);
    vec3 L = normalize(lightVec);
    float NdotL = max(dot(N, L), 0.0);
    vec3 diffuse = lookupTriangleColor(
        eyeHit.idx); // lookup color at
        // the visible point
    gl_FragColor = vec4(diffuse * NdotL,
        1.0);
    return;
}
#ifndef SHADOWS
if(type > 0)
{
    Hit shadowHit = hit;
    if(shadowHit.idx == -1)
        discard;
    gl_FragColor = vec4(-1,-1,-1, 0.0);
    return;
}
#endif
#ifndef RECURSION
{ // else type < 0
    Ray reflRay = ray;
    Hit reflHit = hit;
    if(reflHit.idx == -1)
        discard;
    vec3 reflHitPosition = reflRay.orig +
        reflRay.dir * reflHit.t;
    vec3 lightVec = lightPos -
        reflHitPosition;
    lookupNormal(reflHit, N);
    vec3 L = normalize(lightVec);
    float NdotL = max(dot(N, L), 0.0);
    vec3 diffuse = lookupTriangleColor(
        reflHit.idx);
    gl_FragColor = vec4(diffuse*NdotL
        *0.25 1.0);
}
#endif
}

```

Listing 5: GLSL code for the fragment shader.

Scene (# Δ)	Vertex Shader (%)	Geometry Shader (%)	Fragment Shader (%)
room3 (12)	67.43 / 48.07 / 54.73	1.14 / 5.15 / 5.05	31.51 / 46.79 / 40.21
Cornell Box (36)	85.34 / 77.43 / 72.70	0.50 / 2.47 / 2.68	14.14 / 20.16 / 24.62
Knight in Box (646) (198 animated frames)	99.20 / 98.51 / NA	0.03 / 0.18 / NA	0.76 / 1.32 / NA

Table 1: Computation time distribution between shaders using naive intersection testing. Numbers are for ray casting, shadow casting and shadow casting with single reflections with naive intersection testing (not using the uniform grid). All images are ray traced in 512×512 viewport, all rays are intersecting the scene. GPU instrumentation details are as reported by NVIDIA PerfHUD on a single Geforce 260 GTX, as an average of 50 independent experiments.

	Our method with passthrough GS	Our method without Geometry Shader	⁶ using Fragment Shader
Rays / second (million)	3.322,484	4.890,746	10.485,760
Ray shooting time (ms)	78.9	53.6	25.0

Table 2: Pure ray shooting performance of different setups of the graphics pipeline. All images are ray traced in 512×512 viewport, all rays are intersecting the Torus Knot scene consisting of 1024 triangles. GPU time was computed asynchronously as reported by `GL_EXT_timer_query` on a single Geforce 8600, as an average of 50 independent experiments (by removing the best and worst from 52 independent experiments).

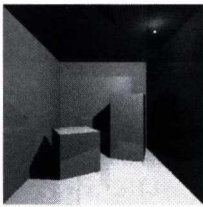






	Cornell Box	Cornell Knot	Dragon	Fairy Forest	Happy Buddha	Horse	Stanford Bunny	
								
# Δ	36	1,024	871,414	174,117	1,087,716	96,966	69,451	
Our Method	RC	64.4	104	481	926	1283	526	295
	SC	213	381	824	1512	1390	714	419
	RT	353	664	2284	2314	1497	878	529
Previous ⁶	RC	17.5	26.8	149	590	777	211	109
	SC	48.5	75.3	327	712	708	312	182
	PRT	119	222	1334	1623	1082	594	381

Table 3: Ray tracing performance in milliseconds. Numbers are for ray casting (RC), shadow casting (SC) and shadow casting with single reflections (RT). All images are ray traced in 512×512 viewport. GPU time was computed asynchronously as reported by `GL_EXT_timer_query` on a single Geforce 8600, as an average of 50 independent experiments (by removing the best and worst from 52 independent experiments).

Real-time foam simulation on the GPU

Tamás Huszár and László Szécsi

hthomas92@gmail.com, szecsi@it.bme.hu
Department of Control Engineering and Information Technology,
Budapest University of Technology and Economics,
Budapest, Hungary

Abstract

Several types of foam can be found both in nature and artificial environments; yet it is rare in computer graphics due to its complex nature. Modelling foam structure and dynamics by simulating the underlying bubble structure has a high computational cost. To model such a complex phenomenon we need to use serious simplifications while maintaining realism and detail.

In this paper we propose a method for rendering dense soap foam in real time. We first build a foam blob from realistic soap bubbles which has a solid inner structure. We use a hybrid method based on ray tracing and 2D billboards to render dense foam constructed from hundreds of these blobs. To model foam behaviour and interaction, we present a simple particle based physics simulation approach. While our method is capable of rendering foam featuring a large number of bubbles, it has certain limitations we also discuss in this paper.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Physically based modeling

1. Introduction

Presenting natural phenomena like smoke, fire or fluids in a realistic way is one of the toughest challenges of computer graphics. Even though the equations describing the physics of these phenomena are known, the exact calculations are too complex to perform in real time. Today's graphics hardware requires some intuitive simplifications or artistic input to efficiently present these phenomena in real-time applications like computer games.

Bubble and foam simulation clearly falls into the above category. The structure of dense foam built of soap bubbles exhibits astounding complexity. While modelling a physically correct soap bubble is an easy task, building complex foam structures from individual bubbles cannot be done in real time on current hardware.

In this paper, we propose a method to render and simulate dense soap bubble foam. We give an intuitive simplification of foam structure which enables us to simulate realistic foam with the speed, detail and quality necessary for real-time applications. After the introduction and previous work, we give a short overview of bubble simulation, discussing the actual techniques used in our method in section 3, including our so-

lution for efficiently modeling multiple connected bubbles. Section 4 introduces the idea of building foam from blobs of soap bubbles. We discuss the structure of these blobs and provide two different methods of storing and rendering blob structure. In section 5, the technique of building actual foam of the blobs is discussed, including a basic physics simulation described in section 6. In the next section we present our result, and provide possible enhancements and future work in further sections, including the conclusion of this paper.

2. Previous Work

Interference phenomena required to understand bubble physics were described by Dias¹. Later, Glassner gave a thorough overview on several aspects of soap bubble physics, including soap film interference and geometric structure of multiple soap bubbles^{3,4}. Most attempts to model bubble and foam structures are offline methods based on ray tracing⁶, and even these offline approaches⁵ use simplified reflection model to render the inner dense parts of the foam to maintain reasonable rendering times.

Recent articles present real-time approaches and use the GPU to simulate bubble formations. Sunkel simulates a

magnitude of hundreds soap bubbles in real time using simplified reflection and lighting model ⁷.

3. Realistic rendering of soap bubbles

3.1. Soap film interference

In order to simulate foam, we must first understand the physics of soap bubbles, and provide an efficient way to render soap films and bubble structures. Basically soap bubbles are gas trapped in a thin fluid layer. Because of surface tension and the inside pressure of the contained gas, the surface of a bubble tends to be minimal. This means soap bubbles can be easily modelled by spheres; this is a common simplification used by most approaches. The interference phenomenon on bubble surfaces can be understood by examining soap film reflection — light interference caused by reflection on two parallel surfaces. Usual soap films are 1–2000 nm wide and have a refraction index of 1.4. Given these values, the intensity change of the reflected light can be calculated by the following equations.

$$ps = \frac{4\pi}{\lambda} nd \cos \vartheta_i$$

$$R_f = 1 - \cos \vartheta_i$$

$$I_r = I_i 4R_f \sin^2 ps$$

The intensity change I_r depends on the incoming intensity I_i , the reflection factor R_f and the phase shift ps . The phase shift can be calculated using the wavelength λ and the incident angle of the light ϑ_i , the index of refraction n and the film width d . The refraction index is calculated using a Fresnel approximation, the film width and the refraction index are constant (we can perturb the film width with random noise to make the bubble more realistic, simulating film thickness changes caused by air pressure variation). By using these simplifications, the intensity change is only dependent on the incident angle and the wavelength, so it can be easily computed or stored in a texture. We used the representative wavelengths of the RGB components, as it is pointless and inefficient to calculate the values over a continuous spectrum. Using these equations and simplifications, a soap film shader can be easily constructed.

3.2. Soap bubble geometry

As we saw earlier, a single soap bubble can be approximated by a sphere. However, for modelling bubble structures, we must compute the shared wall film between the bubbles. Based on Glassner's observations, three soap films always meet at 120° angle and the mutual wall is spherical itself. First, considering two intersecting bubbles, we must determine this auxiliary sphere's centre and radius. The easiest approximation would be a simple planar soap film between the two bubbles. This is an acceptable approximation for distant bubble formations but unrealistic when examined

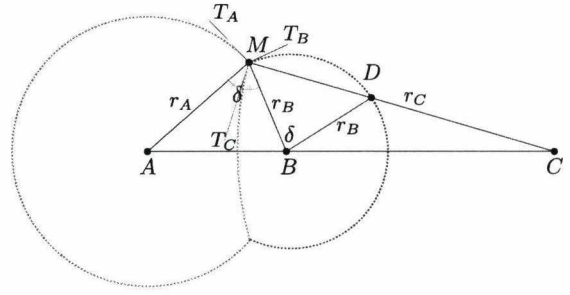


Figure 1: Geometry of bubble walls

closely. In his article, Glassner gives a formula, in which he exploits the aforementioned 120° property. In a general situation without proper physical simulation, when bubbles are spheres of random radii, this rule does not hold. To overcome this we provide a simple but intuitive and visually convincing approximation.

Figure 1 shows the geometry in a 2D slice. We used a simple observation: the tangent is the angle bisector of the angle determined by the intersection of the spheres and the centres ($AMC \angle$). If we extend the bubble model by keeping this rule, but omitting the 120° restriction, we still get acceptable results with reasonable calculation complexity. The formulae given by this assumption are the following.

$$\overline{MD} = r_B \sqrt{2 - 2 \frac{r_A^2 + r_B^2 + \overline{AB}^2}{2r_A r_B}}$$

$$r_c = \frac{r_A \overline{MD}}{r_A - r_B}$$

$$\overline{AC} = \sqrt{r_A^2 + r_C^2 - 2r_A r_C \cos \frac{d+p}{2}}$$

4. Modeling foam using soap bubbles

4.1. Foam structure

While soap foam consists of several soap bubbles, the inner structure of the foam is so complex, that simply modelling it as individual bubbles is not a working solution. Today's real-time methods are capable of rendering hundreds of bubbles, but this is far from the complexity of dense foam. To overcome this, we examined the macrostructure of soap foam. On a large enough scale, below the surface bubbles of dense foams, we see a nearly diffuse and opaque whitish body, hiding the deeper microstructure of the foam. Our idea was to model a foam blob possessing this property. The surface of this blob is built of realistic soap bubbles, and the inside is approximated by an artist-drawn bubble texture representing the inner structure of the foam. This creates the impression of a dense interior with individual transparent bubbles protruding from the blob.

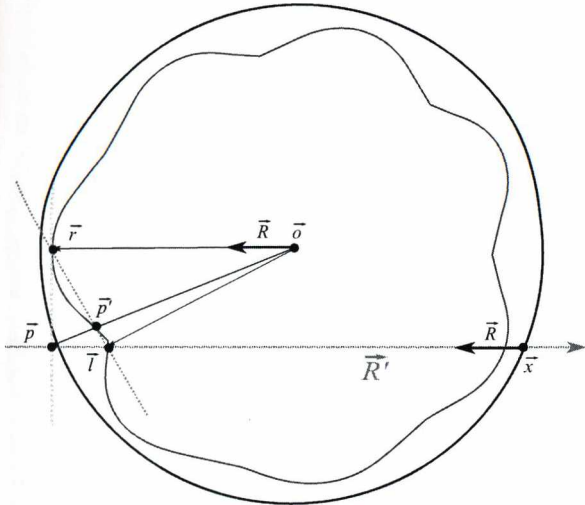


Figure 2: Using the distance impostor technique to model foam blobs

The outer bubbles are spheres drawn using the soap film shader, including the walls between neighbouring bubbles. Due to the complex nature of the blobs, classic polygon-based rendering is not a feasible solution. On the other hand, classic ray tracing is too slow, as complex foam can contain thousands of bubbles. We needed a data structure to store bubble data that makes fast and efficient intersection of the blob and view rays possible.

4.2. Blob intersection using a distance impostor

Our first proposed method is based on the distance impostor technique⁸. The original technique is used for calculating position dependent reflections using environment maps. It stores environment geometry in a cube map, and uses it to iteratively calculate surface points intersected by reflection rays. We imagine the foam blob as an entity trapped in a cube, represented by a cube map storing the distance between the blob surface and the centre in every texel. Now finding the intersection of the ray and the blob is the same problem as the one stated above, with one minor difference: our rays come from outside of the cube map, not from the inside.

As seen in Figure 2, R' is the original ray. We reverse the direction of this ray to get \vec{R} , and place its origin \vec{x} at the second intersection of the original ray and the blob's bounding sphere. As the new ray points towards the near side of the blob (closer to the origin of R'), the iterative search algorithm will find an intersection \vec{l} on this side. Reading a distance value from the side closer to \vec{x} would give a wrong intersection point. Therefore, running the original algorithm with the reversed ray will give the right result.

The required cube map resolution and iteration count de-

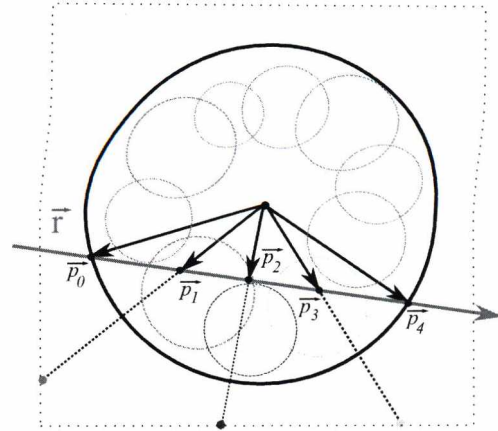


Figure 3: Illustration of the bubble ID technique

pend on the blob geometry. For a common blob consisting of 64 bubbles, which we used for testing, a resolution of 64×64 and a maximal iteration count of 20 were adequate. When using smaller values, visible artifacts appeared near the intersection of the bubbles.

The disadvantage of the method is that we lose the inner structure of the blob as only the outer shell is stored in the texture. As a side effect, this allows the shape of the outer bubbles to be other than spheres. However, when rendering soap bubbles, this does not grant us an advantage, but makes further calculations — like intersection with the inter-bubble wall — impossible. This recognition motivated our second method. Instead of storing distances of the surface, we store the original bubble data, but we heavily reduce the number of necessary intersection calculations.

4.3. Blob intersection by storing bubble identifiers

Once again, we use a cube map to store blob geometry. But instead of distances, we just store an ID of the bubble visible from the outside in the given direction. We also have to store the bubble radii and centres in a separate texture or buffer. First we calculate the intersection of the ray and the bounding sphere, and then we have to find the bubble it intersects first. This would allow us to use an iterative search similar to the one used in the distance impostor technique, but we found that a simple linear search is adequate. This is because finding any texel that contains the ID of the first intersected sphere is sufficient to get an accurate result. As seen in Figure 3, we divide the section of the ray inside the sphere to a fixed number of segments, and then start reading the values from the cube map in the given directions and calculate the intersection of the ray and the corresponding bubble. The first intersection is the one we are looking for.

The advantage of this method is that in most cases, especially if the bubbles are nearly the same size, we will get

the result in the first few iterations. Usually the loop ends in the first iteration when the incident angle is high (the ray goes through the middle of the sphere) and the required iteration count increases as the ray gets further from the centre. Also nothing guarantees that we find the right intersection; in theory we can easily skip the right bubble during the linear search. However experiments showed that when using reasonably sized and evenly distributed bubbles, the results are acceptable. A 128×128 cube texture with 10 iterations produced minor artifacts comparable to the distance impostor technique, and it was also slightly faster.

Storing bubble IDs has another advantage over the distance impostor method: not only the first intersection, but also the intersection with interior walls between bubbles can be computed. When using distance impostors, we only preserve surface geometry, which makes the representation of the precise sub-surface structure impossible. When using the bubble ID technique, we have the exact bubble geometry stored in a buffer. We can use this geometry to calculate inner bubble walls. The linear search algorithm will yield a list of bubble IDs along the ray, if we do not stop it after finding the first intersection. Consecutive bubbles in this list are most likely to form a mutual wall, which can be computed as described in Section 3.2 and intersected with the ray. This is also an approximate solution, as internal bubbles not stored in the ID map are not considered and small bubbles can be skipped by the linear search algorithm. However, with similarly sized bubbles of a soap foam blob this rarely happens, and it does not influence visual quality.

4.4. Inner structure and other details

Using any of the techniques presented above, the rendering of the blob is quite straightforward using ray tracing. We calculate the intersection of the blob and the ray coming from the camera. Then we compute the incident angle of the ray and the surface normal in the previously calculated intersection point. We use these values and an environment map to calculate bubble reflections using the soap film shader. We can also calculate the internal walls for neighbouring bubbles. Finally we have to draw the inner structure using the bubble texture. It should be an artist-drawn image of small bubbles, representing the inner structure. The texture can be stored in another cube map. The sampling direction can be adjusted several ways as it is depending on the given blob and foam type. We used a weighted sum of two vectors. The first vector is the direction of the second intersection of the first intersected bubble relative to the centre of the blob. This spherically projects the texture onto the interior surface of the blob, which is what we get if we remove the outer bubbles. The second vector is the inverse normal of the bubble at the first intersection. This slightly distorts the original mapping based on the outer bubble geometry.

As stated before, the middle of the blob is opaque, while the outer bubbles are nearly transparent. To achieve this ef-

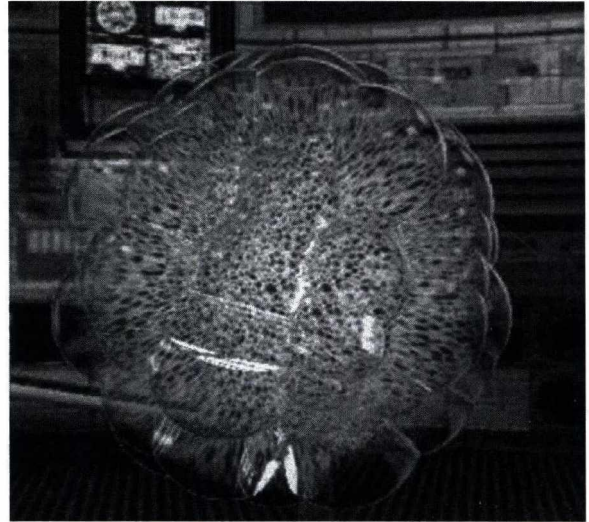


Figure 4: A foam blob rendered using the bubble id method

fect, the transparency of the inner texture is set according to the second intersection point's distance from the inner and outer bounding sphere. In the middle of the blob, the second intersection of the current bubble is closer to the centre as the ray is almost perpendicular to the blob's bounding sphere. Near the outer region, the intersection point is farther from the centre, so the blob will be more transparent there.

The last issue is the surface normal of the blob used for lighting equations. The nearly diffuse inner surface should slightly follow the wrapping surface of the outer bubbles, therefore we used a weighted average of the bubble's normal and the blob's bounding sphere's normal. However this and the other parameters mentioned before should be set according to the actual blob structure and the desired foam type. A typical foam blob can be seen on Figure 4.

5. Rendering foam using foam blobs

Realistic dense foam needs to be constructed from many blobs, so we need a fast technique to render them. While ray tracing the blobs could be straightforward, it would be too slow for large foam. We propose a method based on particle systems, that is capable of rendering hundreds of blobs real time. Blobs are rendered as 2D billboards. The ray from the eye position is calculated for all pixels of the billboard, and it is used to render the corresponding blob as described in the previous section. This means we do not have to do the intersection calculations for all blobs, but also means we cannot calculate inter-blob reflections (which would be too slow to use in real time anyway).

The data associated with the blobs are stored on the graphics card in a vertex buffer. The billboards are generated by the geometry shader using this data. The vertex positions in

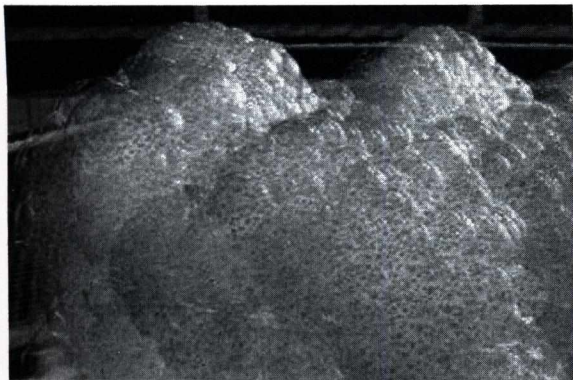


Figure 5: Dense soap foam rendered using the proposed technique

world space are also calculated in the shader, and used in the pixel shader to get the view rays for every pixel. Besides the colour, the depth of the blob is also computed for every pixel to address problems of overlapping particles.

Since the blobs are transparent, we need to sort the particles according to depth, in order to use alpha-blending. However, depth is different in the pixels of the billboards, so we have to do the sorting at pixel level. Depth peeling² is a technique rendering translucent objects. It basically accomplishes pixel level sorting by using multiple rendering passes to store multiple depth levels for every pixel. We use two buffers to store depth data. First we render the scene depth into the first buffer, and then render it again into the second buffer, but only those pixels that are further than the stored depth in the first buffer. Then we flip the two buffers and repeat. We store the blob identifiers in a third buffer, in a different colour channel for each iteration. In the end, the n closest bubble identifiers will be in the final buffer. In the final rendering pass we render the blobs in order with proper transparency. While this method uses multiple rendering passes and it is generally slow, it provides a real-time alternative to ray tracing.

This method has one serious shortcoming in cases where multiple blobs overlap each other. Let's assume that we use three rendering passes (and store 3 layers of blob depths). Now imagine that the first three blobs are rather transparent, but there is a fourth, solid blob behind them. In this scenario, the resulting foam would be transparent, resulting a transparent hole in the foam. To overcome this limitation, we use an extra rendering pass to calculate the maximal opaqueness for all visible blobs. When we draw the diffuse foam texture, we use this value to plug the unwanted holes in the foam. The main disadvantage of this method is the resource consumption, as we have to calculate intersection points for all blobs (however we do not calculate surface interference, reflections or wall geometry for these blobs). Figure 5 shows

dense foam. All the free parameters were set to grant visually appealing result.

It was not stated explicitly before, but blob rendering techniques require the blob structure to be static. The blob texture is prerendered once and then used for all blobs. This means that all the blobs are the same (it is possible to use several blob textures to construct a fixed number of different blobs). We used transformations to change blob size and orientation. This can be easily done real time in the pixel shader during the intersection calculations and grants us more diverse foam. To store these transformations, only one additional float vector is required in the vertex buffer. We can represent the orientation as a quaternion and store it in a four-dimensional vector. The blob size can be the fourth, previously unused coordinate of the position vector. We can also use a transformation matrix to store more general affine transformations, but in our implementation it was unnecessary to do so.

6. Foam physics

To provide even the most basic physical simulation, we must render solid objects beside the foam. As we used environment mapping for reflections and refractions, solid objects must be rendered using a blending technique. We first render these objects, and then blend the foam over them without clearing the depth buffer. This is efficient but this does not handle reflections of solid object on bubbles, or the rendering of transparent objects like smoke or glass.

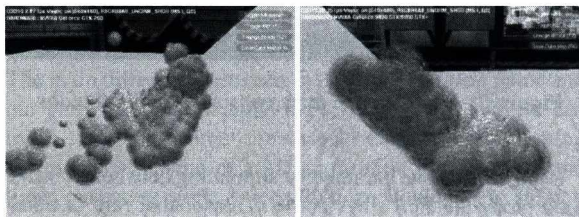


Figure 6: Foam formations sliding down on a slope

To simulate foam dynamics and present the characteristics of this rendering technique we created a basic but fast physical simulation based on particle dynamics. The simulation is able to handle inter-particle forces and outside objects. In the technical demonstration we presented a foam mass sliding down on a slope (see Figure 6). Each blob has a position, mass, and velocity, with forces acting between blobs and other objects. The blobs are represented by their bounding spheres. If two blobs get too close, two types of force can affect them: if they are far enough an attractive elastic force arises, modelling the different parts of the foam sticking together. However, if two blobs get too close, they collide, resisting collapse and giving the foam a solid structure. By properly adjusting these forces and the gravity, a good approximation can be achieved for the desired foam type.

The simulation is done on the CPU. Further exploration in this topic could yield more realistic results and boost performance by implementing a physical simulation on the graphics card.

7. Results

We implemented the technique using DirectX 10 and Shader Model 4.0 on an NVIDIA GeForce GTX 260 graphics card. We achieved real-time simulation (32 FPS) of 100 blobs and a total number of 22700 separate bubbles, using the bubble ID technique for calculating intersections. The various parameters like iteration count, texture size, and the number of bubbles in a blob were set to imitate soap foam to the highest possible fidelity without visible graphical glitches. Further tweaking these parameters could result in performance increase, while maintaining acceptable graphic quality.

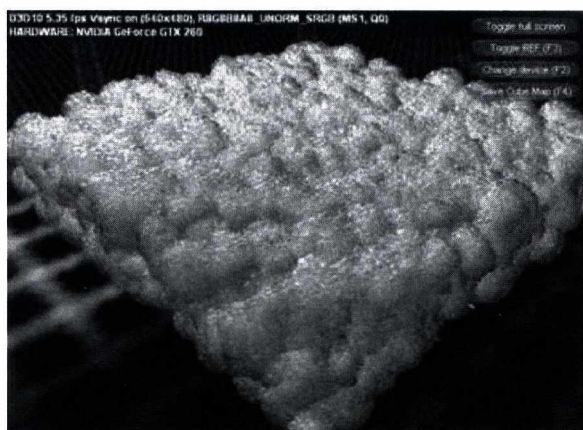


Figure 7: A box shaped form consisting of 5000 blobs

Using these same parameters, simulating between 100 and 1000 blobs the FPS stays above a reasonable rate (around 10). Figure 7 shows a foam formation of 5000 blobs and a total number of 1 135 000 bubbles, rendered at 5 FPS. Given these numbers, in the near future with further optimisations the real-time simulation of foam consisting of hundreds of thousands of bubbles could be possible.

8. Future work

The most serious limitation of the proposed technique derives from the particle based approach. Our blobs are static entities with fixed size and structure, and when we start to divide the foam into smaller pieces comparable to the blob size, it leads to artificial and unnatural results. To overcome this, we propose a possible direction. First, we need a model to adjust blob size dynamically, based on some physical observations, but not too complex to undermine the performance. Another possible way is to locally increase and decrease the simulation resolution (the blob size in this case)

by the local foam characteristics and viewer distance. Blobs inside a dense foam or far from the viewer could be merged together, or their simulation and render quality should be otherwise decreased, while larger blobs broken out of the foam should break up into smaller blobs.

9. Conclusion

Bubbles and foam are extremely complex natural phenomena, the formation, motion and optics of which obey complex physical laws practically impossible to simulate in real time. A visually convincing result, however, is feasible with clever data structures and subtle approximations. As with a wide range of natural geometries, the concept of impostors is very helpful. We have shown how a generic impostor technique – the distance impostors – can be modified to represent bubble clusters, and we also proposed a specialized representation that exploits the fact that bubbles are spherical, and allows not only for a more accurate representation, but also for an approximation of internal foam walls. Furthermore, we described algorithms for the simulation and rendering of massive foam composed of the bubble clusters, based on particle systems and the billboard visualisation technique. Our method is capable of real-time rendering dense foam consisting of ten thousands of bubbles on modern graphics hardware.

References

1. L.M. Dias. Ray tracing interference color. *IEEE Computer Graphics and Applications*, 11(2):54–60, 1991.
2. C. Everitt. Interactive order-independent transparency. *White paper, nVIDIA*, 2(6):7, 2001.
3. A. Glassner. Soap bubbles: Part 1. *IEEE Computer Graphics and Applications*, 20(5):76–84, 2000.
4. A. Glassner. Soap bubbles: Part 2. *IEEE Computer Graphics and Applications*, 20(6):99–109, 2000.
5. S. Rosenbaum and M. Bergbom. Foam. <http://cs.stanford.edu/people/rosenbas/foam>, 2007.
6. Y. Sun, F. D. Fracchina, T. W. Calvert, and M. S. Draw. Deriving spectra from colors and rendering interference. *IEEE Computer Graphics and Applications*, 19(4):61–67, 1999.
7. M. Sunkel, J. Kautz, and H.-P. Seidel. Rendering and simulation of liquid foams. In *Vision, Modelling and Visualization*, 2004.
8. L. Szirmay-Kalos, B. Aszodi, I. Lazanyi, and M. Premecz. Approximate ray-tracing on the gpu with distance impostors. *Computer Graphics Forum*, 24(3):695–704, 2005.

Graphics Techniques in the Turing Game Project

Tamás Umenhoffer¹

¹ Department of Control Engineering and Information, Budapest University of Technology and Economics, Budapest, Hungary

Abstract

This paper covers the graphical development of the test applications created for the Turing Game project. We present our rendering system including lighting, post processing, animation and camera setup. We also describe the non-photorealistic techniques we used to achieve stylistic rendering.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Color, shading, shadowing, and texture, Animation, K.8.0 [Personal Computing]: Games

1. Introduction

The Turing Game project is a research project that deals with artificial intelligence, human-computer interaction and human-computer collaboration. The main goal is to create a game-like application where human and computer players act together in a virtual world, they have tasks to do and these tasks usually need some kind of collaboration and thus communication. Human and computer players should use the same language for communication, preferably they should form a new language from a set of predefined signs by assigning meaning to them and mixing them together.

The first test environment was a simple game based on a the popular *PacMan* game. This game is a multiplayer game and allowed only human players. Human players are pacmans and the goal of the game is to survive in a labyrinth-like environment where ghosts are chasing the pacmans. Pacmans can neutralize ghosts with picking up power items or encircle a ghost. Communication is made with gestures (pointing and special actions), and facial expressions.

In this paper we describe the graphical techniques used in this test game. We used an open source 3D rendering engine called OGRE² and NVidia's free physical simulation engine called PhysX¹. All development was made in C++, and run on Windows platform using DirectX 9 and HLSL. The game has to offer high frame rates as it should run on middle class hardware too, which made us concentrate on rendering techniques that has low computational costs. In the next sections first we introduce our lighting system, then our stylistic ren-

dering techniques. This is followed by the description of our post processing effects and animation system. The last sections cover collision detection and camera setup.

2. Lighting and shading

The main difference between the original *PacMan* game and our game is that our game uses 3D graphics; players can move freely in a three dimensional environment. Creating the graphical engine of a 3D application is a complex and time consuming work, especially if we would like to achieve a modern look featuring up-to-date rendering techniques. This is why we decided to use an open source rendering engine that has all these features implemented. OGRE is a constantly developed high quality engine that has a wide community making it an ideal choice for us.

OGRE has a flexible, scriptable material system supporting all formats of GPU shader programs we could use to implement our lighting methods. For static level geometry we used precalculated lighting stored in light maps. These maps were precalculated and saved to texture files by the 3D modeler we used. Light maps usually have low resolution which cannot capture the lighting changes due to high frequency geometry features. In our case these features were modelled with normal maps, which can be used in light maps too. We modulate the radiance stored in the light maps with the cosine of the angle between the triangle normal and the perturbed surface normal read from the normal map. Thus we consider all precalculated irradiance as it would arrive

from the triangle normal direction. Note that a more precise solution can be achieved with storing several light maps each storing incoming radiance from a given direction and combine them during rendering⁶. This method can also support light mapping for specular materials, but needs special tools to calculate these light maps. This technique supports only static lighting, thus for the dynamic lights present in the scene (which were also used for communication purposes) we added their contribution with per-pixel Phong-Blinn lighting.

For dynamic objects we needed a different approach. During lighting the static geometry we used a lot of light sources which did not affect final rendering time due to the use of precalculated lighting in light maps. However, for dynamic objects only a few light sources can be taken into account to maintain high frame rates. Though we could select a few significant light sources, or take into account only the closest ones (OGRE supports this automatically), we used an even simpler solution. We treated these light sources as environmental light sources and also used precalculation for them. The most obvious way would be the use of environment mapping with which high quality indirect lighting effects can be achieved⁹. However, this requires a frequent update of an environment map (or even a distance impostor) or interpolation between several precalculated environment maps. Interpolated environment maps would produce satisfying frame rates, but this technique would be hard to automate with our tools, and OGRE also does not support it automatically, so we used an even simpler solution.

Lighting of dynamic objects was handled with a special two dimensional light map texture that was calculated for a diffuse white plane placed half of character size above the ground. For any shaded point of a dynamic object we can calculate light map coordinates from its world space coordinates and use the stored light intensity for static lighting. This method is only an approximation, but since in our scene the characters are moved on a plane and light sources are placed relatively high above the character the result was satisfying and had really low computational costs. Figure 1 illustrates this lighting technique with a box placed at different positions in the virtual world. The effect of dynamic light is added to precalculated lighting just like in case of level geometry.

The effect of dynamic objects on static geometry is usually handled with some shadow computation method. We used the built-in shadow volume feature of OGRE which creates high quality sharp shadows. In our case we again run into the problem of having too many light sources. We should have one or two significant light sources for shadow computation, but these are hard to select. Thus we created a separate light source dedicated to shadow computation. This light source does not give any light contribution to the shaded scene, it only produces shadows, and it is placed high above the center of the level. Again as our static light sources

are placed high above the character this approximation is not annoying, but shadows help us a lot in the interpretation of the three dimensional space.

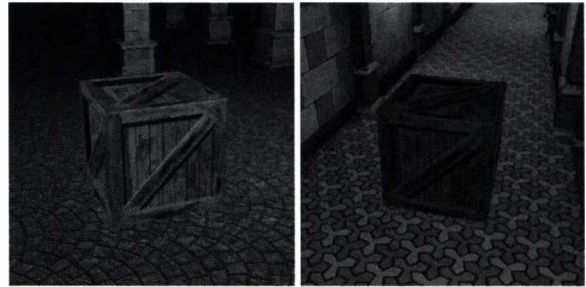


Figure 1: Precalculated lighting of a dynamic box object.

3. Stylistic rendering

Though we used realistic lighting, we wanted to give a cartoon-like look to the game. Stylistic rendering can give an extraordinary look, which catches the eye and inspires the user to try the game. Further more using an abstracted rendering style some communication features like facial expressions are easier to interpret than in a realistically shaded lighting.

For static level geometry we used simplified textures with large area features to provide a sufficient background for dynamic objects rendered with special render techniques. As textures are only used for level geometry we saved a lot of texturing and UV mapping work in case of our character models.

3.1. Cell shading

For our player characters we used a technique similar to cell shading. Cell shading mimics traditional cartoon painting with mapping smooth lighting values to discrete shades to create the characteristic flat look. In practice we calculate the luminance value of total outgoing radiance of the shaded point (assuming that the surface color is white) and map this scalar value with a gradient map created by the modeler artist. This gradient can have both smooth and sharp transitions, but for a flat look sharper transitions are preferred. Figure 2 shows two objects rendered with the described cell shading method and their gradient maps.

3.2. Contour detection

Cartoon paintings have dark outlines and contour lines which should also be rendered. These lines can be generated from the triangle mesh emphasizing the edges where the sign of the dot product of the view direction and the normal vector of two adjacent triangles changes. We can also take into account edges which became contour edges with

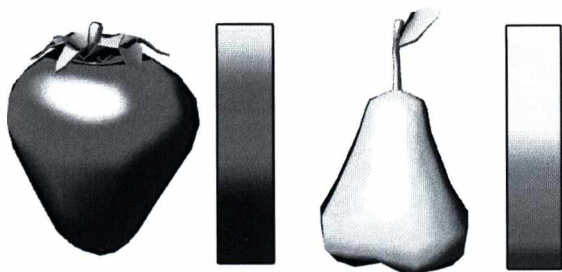


Figure 2: Fruits rendered with cell shading and their gradient map.

a little perturbation of the view direction (these are called suggestive contours³). This technique requires huge amount of CPU processing in each frame, so a more GPU friendly and lighter method should be used. The dot product of the view vector and the shaded normal can be determined eg. by the fragment shader in each pixel. Where this dot product is smaller than a threshold value the pixel contains a contour line. This technique gives satisfying results only if the geometry is well tessellated (see Figure 3) which has a great impact on performance, so we used a different method and used image-processing for outlining.



Figure 3: Silhouette detection using the facing ratio (the dot product of the view vector and the shaded normal) of shaded points. The cherry model on the right is highly tessellated which results in higher quality contours.

We used a method similar to Decaudin's method⁴ to detect outlines from per pixel normals and depth buffer. Silhouette edges can be thought of as discontinuities in the depth buffer, while crease edges are discontinuities of the normal vectors. To detect these discontinuities per pixel normals and depth values should be stored and some kind of edge detection filter should be applied to them. We tried several filter types and filter sizes to find the balance between performance and image quality and finally we used a difference of Gaussian filter with setting the standard deviation of one of the Gaussians to zero (which means the original image). The depth buffer and normal buffer were stored in a single texture so Gaussian filtering can be performed parallel on the depth

values and the x and y components of the normal vectors. The filter size and contribution of the depth value and normal vector differences should be carefully adjusted to get the desired effect. The final difference image is inverted and multiplied with the rendered image. Figure 4 shows the inverted difference images of depth values and normal vectors and the final composited rendering (Figure 5 shows screen shots taken from the final game using stylistic rendering).

As distance objects show aliasing artifacts and their outlines should be less intensive as for nearby objects they are lightened proportional to camera distance. To have thinner outlines for far objects filter size can also be adjusted according to camera distance. To further reduce aliasing artifacts the normal and depth buffers should be rendered using doubled resolution and aliasing will be automatically performed by hardware texture filtering.

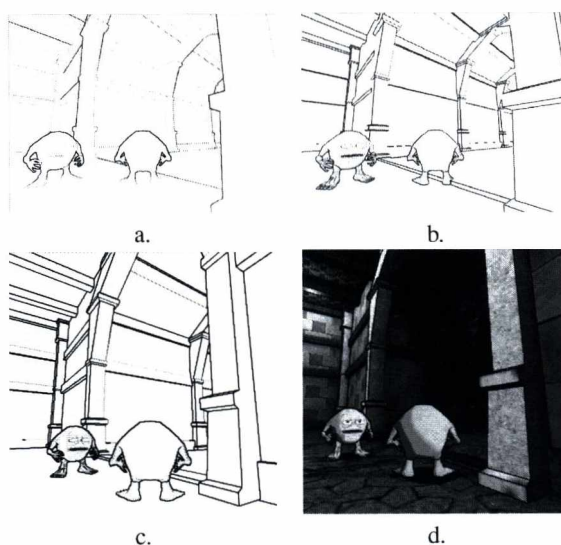


Figure 4: Contour detection with image processing. a.: contours detected from the depth buffer, b.: contours detected from the surface normals, c.: final contour image combining a. and b., c.: final composited image.

3.3. Edge preserve filtering

Beside stylistic rendering we tried a different approach to achieve a stylistic look: we rendered the scene with realistic shading and used image processing with special filters on the rendered image. Our implementation is based on the work of Kyprianidis and Döllner⁵. Automatic image abstraction is done with an orientation-aligned separable bilateral filtering and a final color quantization provides the cartoon-like look. This approach aligns the bilateral filter with the local structure of the image, thus the separable implementation of the filter first filters in the direction of the gradient and then

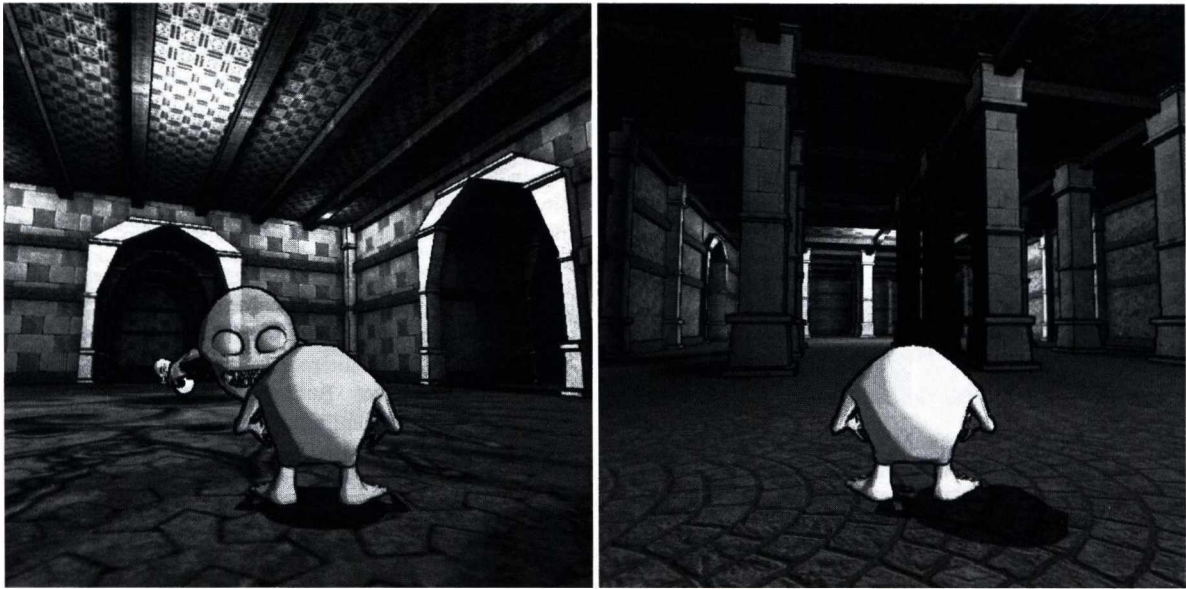


Figure 5: Screenshots from the application.

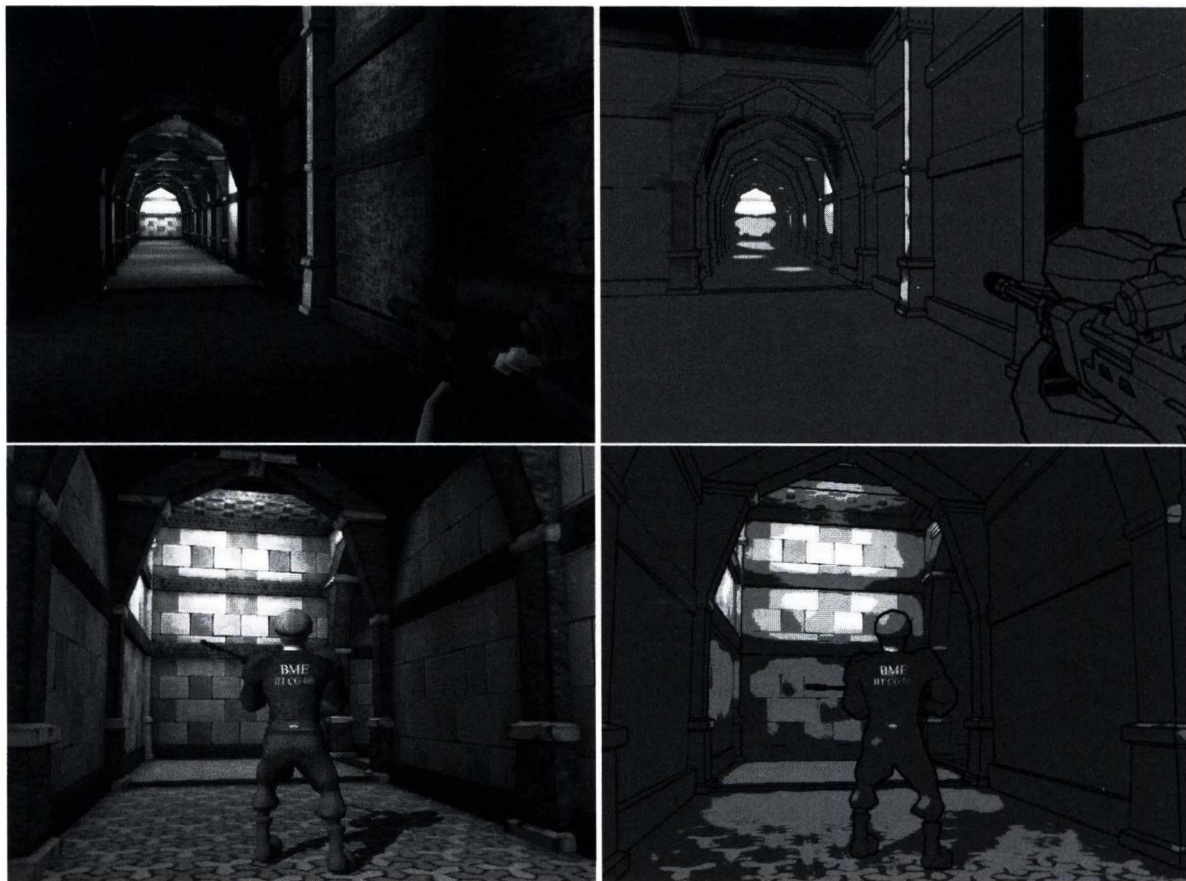


Figure 6: Edge preserve filtering. Left: original rendered image. Right: image after filtering and contour detection.

in the direction of the tangent. This approach eliminates the horizontal and vertical artifacts of the original xy -separable bilateral filter⁷. To construct the tangent field first a Sobel filter is used to approximate the derivatives, then the structure tensor is computed, which is smoothed with a separable Gaussian filter and finally the tangents are computed from the smoothed tensor. To avoid color bleeding artifacts the bilateral filtering is performed in $Y'CbCr$ color space, which also makes color quantization easier. We used the same contour detection technique as described in 3.2.

The strength of this technique is the use of separable filters only which provides good performance. OGRE also has a good scriptable post processing framework to support these image processing tasks. However the computational cost is still too high for some configurations, and the complexity of the shaders also required Shader Model 3 GPUs, which could not be a minimum requirement for the game. For these reasons this technique remain an interesting but experimental only feature. Figure 6 shows the results of the edge preserve abstraction filtering technique.

4. Post processing effects

OGRE's post-processing framework, and image composition helped a lot to create some special effects of the game. One of these effects was the "visualization of the power". When a player picks up a power item we should indicate somehow that this player has gained some special abilities and can neutralize ghosts for a sort time. We wanted a colorful swirling aura to appear around these characters as well as around the power item itself. We visualized a force field similarly between two players who got close enough to each other and can encircle and neutralize the enemy together.

To draw this aura first we identify the objects that need this aura and draw them to a separate image with an animated color texture. Depth values should be checked with the depth buffer of the entire scene. We render the force field as a cylinder. This image is then smoothed with a separated Gaussian filter, and finally added to the original image. Figure 7 illustrates the compositional method to visualize the power auras.

We also added a depth of field effect with post-processing, which blurs objects that are out of focus with a depth based Gaussian filter. We set the focus at a fixed distance, the distance of the player's character. In our Gaussian filters we used importance sampling to enhance quality⁸.

5. Animation

Character animation plays an important role in our game as it is the only way of communication between players. Players can point at something in 3D space or can do special actions. These were created with skeletal animation and skinning. Characters can also express basic feelings with facial expressions which were implemented with vertex blending.

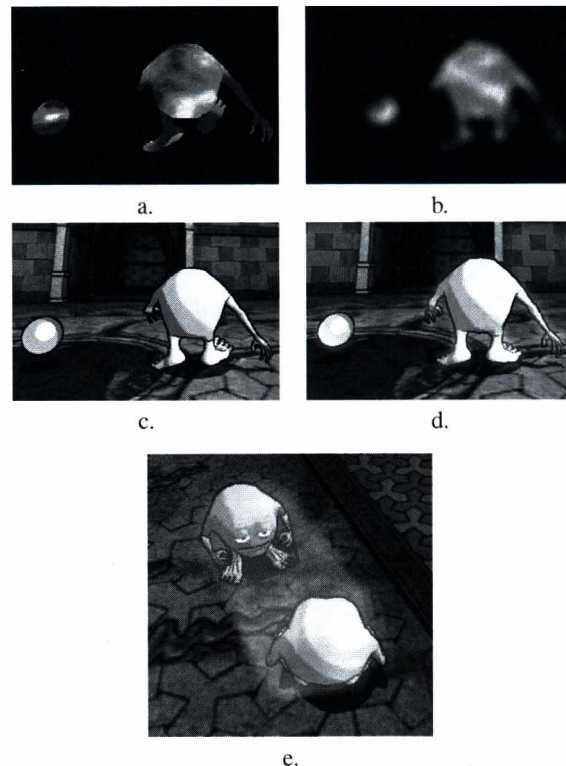


Figure 7: Rendering steps of the power aura visualization. *a.:* objects with power rendered to an image, *b.:* blurred power image, *c.:* original rendering, *d.:* rendering with power visualization, *e.:* visualizing the force field between two characters.

5.1. Skeletal animation

Skeletal animation and vertex skinning are well supported by the OGRE engine. Keyframe skeleton animations can be saved from the most common modeler programs and can be played by the engine automatically. Software vertex skinning is also automatically performed on the CPU, but hardware skinning can also be used by an appropriate GPU shader program.

Pointing in an arbitrary direction with the character's hands can be solved with inverse kinematics, but unfortunately this is not implemented in OGRE so we used a simpler method. Instead of solving the equation system of inverse kinematics we used forward kinematics and animation blending. In animation blending (which is a built in feature of OGRE) several animations of the same skeleton are played simultaneously and their effects can be added together with weighting. To implement pointing we created poses that point to the main directions of the 3D space (Figure 8), and in runtime we combine three of these main poses with proper weighting for a given pointing direction. The

final result will be correct and requires negligible computational power. On figure 9 a green double arrow, which is controlled by a pointing device (which can be mouse, a game pad or a Wii controller), denotes the point where the character points. To reconstruct this point in 3D space we use the mouse screen coordinates, the depth buffer and the inverse of the camera transform.

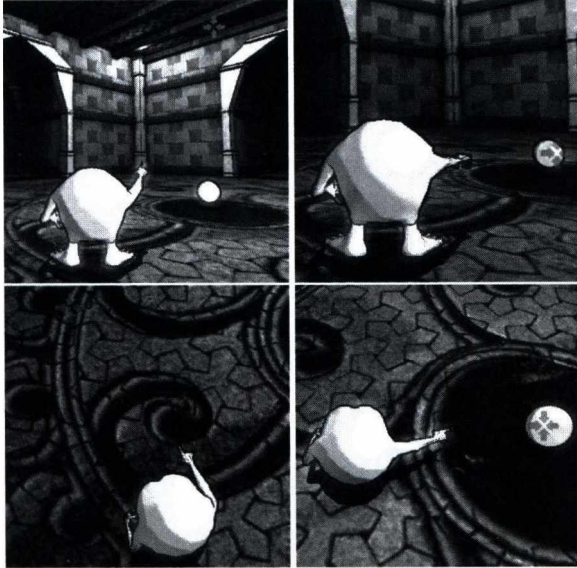


Figure 9: Pointing with a character in the game. The main poses are blended together to create an arbitrary pointing direction.

5.2. Facial animation

Facial expressions are the second way of communication in our game. For example the player can express his satisfaction or disappointment about the actions of another player. The most common way to create facial animation in games is to use skeleton animation or to use blend shapes. We used both techniques. We created and animated a bone for the chin and used blend shapes to mimic the effect of facial muscle activity.

We created four main expressions which can be mixed together: happiness, sadness, anger and fear (figure 10). These expressions can be weighted and added together automatically by OGRE (figure 11 shows some mixed expressions of our character model). These expressions can be controlled by buttons, however in this case defining the mixing ratio is problematic, so only one expression will be used at a time. Another solution is to create some user interface component with which mixed expressions can be easily set, or expressions can even be detected from a video sequence taken from a webcam. Our final implementation uses only one expression at a time and uses buttons to activate them. Facial ex-

pression detection was also supported with the help of an external library called FaceApi. Using discrete expressions did not prove to be a disadvantage as expression mixing does not play a great role in fast communication.

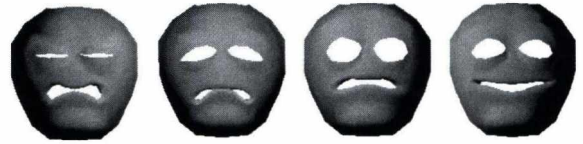


Figure 10: Face models of the main expressions. From left to right: anger, sadness, fear and happiness.



Figure 11: Facial expressions created with blending the main expressions.

6. Collision detection

Out game characters can move freely in a three dimensional environment. To prevent characters penetrating into each other and the scene geometry collision detection and handling is needed. Unfortunately OGRE does not implement collision detection, so we used PhysX to handle collisions. The level geometry was approximated with boxes, and the characters were approximated as capsule shapes. We created a script for the modeler we used to export the collision boxes of the scene geometry (which were defined by the modeler artist) into a text file, and the application builds up the physical representation from this file.

Camera movement also needed some special considerations. The game uses a third person tracking camera setup, which places the camera behind and slightly above the player character. This camera setup can be disturbing when the player gets occluded by other objects, this usually happens if the player is close to walls for example. To overcome this we used PhysX's features (ray object collision testing) to detect if the player is occluded by an object (typically by walls in our case) viewed from the camera, and move the camera closer and upper in front of the occluding object. Though this limits camera placement, it totally eliminates occlusion artifacts which is much more disturbing.

7. Conclusions

A modern game should have high quality graphics and should use high performance techniques to provide real-time

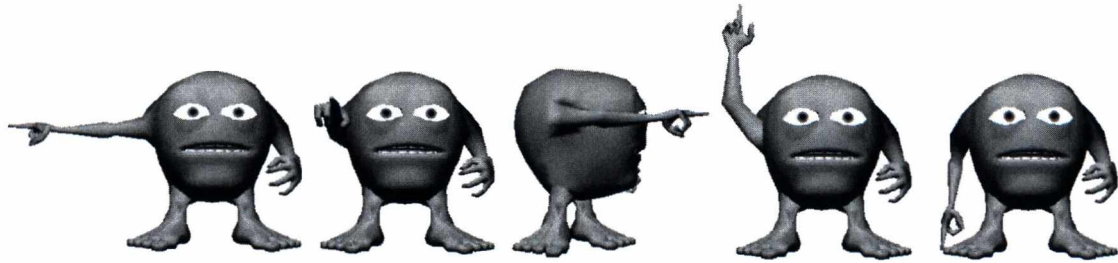


Figure 8: Character poses for the main pointing directions.

frame rates on various platforms. On the other hand graphics programming can be very complex and cover various areas like lighting and rendering, animation and physics. For a small game developing project the most efficient choice is to use open source or free frameworks to save developing time. OGRE and PhysX proved to be a good choice for us, they provide enough capabilities to implement our rendering techniques without writing too much engine code. Stylistic rendering also proved to be a good choice as it saved us a lot of digital content creating work and also gave a unique look. We used techniques that can easily be implemented without writing special preprocessing tools, runs at high frame rates but still provide high quality, visually pleasing result.

Acknowledgements

This work has been supported by the Turing Game Project sponsored by the United States Air Force and OTKA K-719922 (Hungary). The Turing Game Project is led by the Neural Information Processing Group at the Department of Information Systems, Eötvös Loránd University.

References

1. Nvidia physx official home page, http://www.nvidia.com/object/physx_new.html.
2. OGRE official home page, <http://www.ogre3d.org>.
3. Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape, 2003.
4. Philippe Decaudin. Cartoon looking rendering of 3D scenes. Research Report 2919, INRIA, June 1996.
5. Jan Eric Kyprianidis and Jürgen Döllner. Real-time image abstraction by directed filtering. In W. Engel, editor, *ShaderX7 - Advanced Rendering Techniques*, pages 285–302. Charles River Media, 2009.
6. Jason Mitchell, Gary McTaggart, and Chris Green. Shading in valve's source engine. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, pages 129–142, New York, NY, USA, 2006. ACM.
7. T.Q. Pham and L.J. van Vliet. Separable bilateral filtering for fast video preprocessing. *IEEE International Conference on Multimedia and Expo*, 0:4 pp., 2005.
8. Balázs Tóth, László Szirmay-Kalos, and Tamás Umenhoffer. Efficient post-processing with importance sampling. In W. Engel, editor, *ShaderX7 - Advanced Rendering Techniques*, pages 259–276. Charles River Media, 2009.
9. Tamás Umenhoffer and László Szirmay-Kalos. Robust diffuse final gathering on the gpu. In *Winter School of Computer Graphics '07*, pages 121–129, Plzen, Czech Republic, 2007.

Shadow map filtering for rendering volumetric phenomena

Viktor Heisenberger and László Szécsi

hv647@hszk.bme.hu, szecsi@iit.bme.hu
Department of Control Engineering and Information Technology,
Budapest University of Technology and Economics,
Budapest, Hungary

Abstract

When rendering lighting effects of homogenous participating media with single scattering, like light shafts or sun rays, the visibility of the light source is the most important factor determining the appearance of the image. In real-time applications, this visibility term is typically found using depth shadow maps. As a depth shadow map is a sampled representation of the shadowing geometry with a finite resolution, aliasing artifacts emerge. For surface shading applications, numerous shadow map filtering techniques have been proposed, including both post-filtering (most notably percentage closer filtering) and pre-filtering approaches (e.g. variance shadow maps). However, when rendering volumetric phenomena, the shadow map is sampled in a characteristically different way. In this paper, we investigate the properties of the radiative transport equation with respect to visibility sampling, and propose pre-filtering techniques for shadow maps, along with the modified ray marching algorithms that exploit the extra information stored in the processed maps. These achieve similar image quality with less samples and with better frame rates.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1. Introduction

Rendering volumetric light transport phenomena, or participating media, has become one of the main challenges in real time graphics. As the processing power of graphics hardware now allows for a convincingly accurate simulation of solid surface reflection effects, even under complex lighting conditions and various material properties, it is also possible handle more subtle volumetric effects for increased realism. At the same time, the computation of these effects is inherently more complex. To find the color of a pixel, it is not enough to locate a single visible surface point, but we must aggregate light scattered towards the eye from every point along the ray. How much light arrives at these points also depends on the participating media.

In this paper, we examine the use of shadow mapping to determine the visibility of light sources while rendering participating media. To this end, we do not consider volumetric effects not closely related to shadows. Specifically, as it will be discussed in detail in Section 1.1, we neglect emission, multiple scattering and inhomogeneities of the medium. The

effects rendered under these assumptions are usually known as light shafts or sun rays, and they are among the most spectacular and ubiquitous features in real-time applications.

1.1. Radiative transport in participating media

There are four phenomena that can influence the radiance travelling towards the eye along the ray: absorption, out-scattering, in-scattering and emission. In this paper, we do not discuss emission effects. Photons originating from light sources may collide with the particles of the participating medium. The probability of this happening over a path of unit length is described by the optical density τ . After a collision, the photon may be absorbed or scattered, the latter happening with probability a , which is called the *albedo*. The medium is *homogenous* if the density and albedo do not depend on the location. The probability density of the scattering direction is defined by the *phase function* $P(\omega', \omega)$, where ω' is the incoming and ω is the outgoing direction.

Let us consider a ray of equation $\vec{x}(s) = \vec{x}_0 + \vec{\omega}s$, where \vec{x}_0

if the ray origin, $\vec{\omega}$ is the ray direction ω expressed as a unit vector, and s is the ray parameter in range $[0, S]$. It is also true that $\vec{x}(S)$ is the eye point, \vec{x}_0 is the surface point hit by the ray, and S is the distance between the two. The medium behind the visible surface point, where $s < 0$, cannot contribute to the radiance at the eye.

The change of radiance $L(\vec{x}, \omega)$ along this ray in non-emissive homogeneous participating media is expressed by the radiative transport equation⁸

$$\frac{dL(\vec{x}(s), \omega)}{ds} = -\tau(\vec{x}(s))L(\vec{x}(s), \omega) + \tau(\vec{x}(s))a(\vec{x}(s)) \int_{\Omega'} L(\vec{x}(s), \omega') P(\omega', \omega) d\omega'$$

This integro-differential equation is difficult to solve since the unknown radiance appears in derivative, normal, and integrated forms. Such equations can be solved by Monte Carlo methods⁷, but they are far too slow for real-time applications.

However, there is a wide class of participating media scenarios where significant simplifications can be made, and an analytic solution can be obtained. Firstly, density τ and albedo a can be considered constant if the medium fills all empty space evenly. This is true in the case of dust hovering in the air, water in an underwater environment, or fog outdoors. If the density is also low, then indirect, scattered light arriving at a point is negligible compared to direct lighting. Therefore, multiple scattering can be ignored. These two assumptions mean that every considered light path consists of two segments passing through a homogenous medium: one from the light source to the point of scattering, and one from there to the eye.

Furthermore, real-time applications usually work with the abstract light source model. That is, directional and point light sources are assumed, and the direct illumination can be found by evaluating a form factor analytically. The visibility of the light sources is typically determined using shadow maps. For the simplicity of discussion, we assume a single abstract light source. Multiple light sources can be handled by summing the contributions of individual light sources.

With the above assumptions, the in-scattering term

$$\int_{\Omega'} L(\vec{x}(s), \omega') P(\omega', \omega) d\omega'$$

can be substituted with

$$L_{\text{light}}(\vec{x}(s), \omega') v(\vec{x}(s)) P(\omega_{\text{light}}, \omega),$$

where L_{light} is the unoccluded direct incoming illumination due to the abstract light source, which does not depend on scene radiance or geometry. Function v is the binary visibility factor. The directional integral has been eliminated because $L(\vec{x}(s), \omega')$ is a Dirac-delta function, which is zero everywhere except for ω_{light} , the direction of the light source.

Thus we get

$$\frac{dL(\vec{x}(s), \omega)}{ds} = -\tau L(\vec{x}(s), \omega) + \tau a L_{\text{light}}(\vec{x}(s), \omega_{\text{light}}) v(\vec{x}(s)) P(\omega_{\text{light}}, \omega). \quad (1)$$

In such a low-density, evenly distributed medium, volumetric scattering produces a high variation pattern in the image only if the lighting also has a high variation. That is, there should be dark, shadowed parts and brightly illuminated regions. This typically happens when strong light arrives into a restricted area of a relatively dark environment. In this scenario the visibility v of the light source becomes the most important factor. Therefore, the accuracy of the shadow mapping technique used to determine light source visibility is the most critical in these cases. For this reason, when investigating shadow map filtering techniques for rendering volumetric phenomena, we choose the problem of single scattering in a homogeneous medium as the test environment. In immersive real-time applications, this phenomenon manifests as light shafts or sun rays.

The differential equation 1 can be solved analytically (the correctness of the solution can be proven by inserting it back into Equation 1):

$$L(\vec{x}(s), \omega) = e^{-\tau s} L(\vec{x}_0, \omega) + \int_0^s L_{\text{light}}(\vec{x}(s), \omega_{\text{light}}) v(\vec{x}(s)) P(\omega_{\text{light}}, \omega) e^{-\tau(S-s)} ds. \quad (2)$$

The integral on the right hand side of the equation can be approximated with a finite Riemann summation

$$L(\vec{x}(s), \omega) \approx e^{-\tau s} L(\vec{x}_0, \omega) + \sum_{n=0}^{N-1} L_{\text{light}}(\vec{y}_n, \omega_{\text{light}}) v(\vec{y}_n) P(\omega_{\text{light}}, \omega) e^{-\tau n \Delta s} \Delta s, \quad (3)$$

where the step size is $\Delta s = S/N$, i.e. it is the length of the ray divided by the number of sample points $[\vec{y}_0, \dots, \vec{y}_{N-1}]$. Sample points are located on the ray at

$$\vec{y}_n = \vec{x}_0 + (N - n) \Delta s.$$

Note that we chose the indexing of the summation to start near the eye and increase with the distance.

The *Henyey-Greenstein* model is a widely used approximation for the phase function $P(\omega', \omega)$:

$$P_{\text{HG}}(\omega', \omega) = \frac{1}{4\pi} \frac{3(1-g^2) \left(1 + (\vec{\omega} \cdot \vec{\omega}')^2\right)}{2(2+g^2) \left(1+g^2-2g(\vec{\omega} \cdot \vec{\omega}')\right)^{\frac{3}{2}}},$$

where g is the model parameter called *anisotropy* which defines the average cosine of the scattering angle. Dust, fog, water, or similar phenomena exhibit very little anisotropy;

as the viewpoint moves, we do not expect a change in light intensity. With $g = 0$, we get the Rayleigh phase function

$$P_R(\omega', \omega) = \frac{3}{16\pi} (1 + (\omega \cdot \omega')^2).$$

This is the phase function we used in our implementation.

1.2. Shadow maps

Shadow mapping¹⁰ is the most prevalent way of rendering shadows in real time. It is a two-pass method. First the shadowing geometry is rendered into a shadow map texture so that the surface that is visible from the lightsource is projected onto the shadow map texels, also called *lexels*. Every lexel stores information about the position of the surface element, typically in the form of a depth coordinate or distance from the lightsource. Together with the shadow map texture coordinates of the lexel, these are enough to encode the 3D position. Thus, the shadow map is a sampled representation of shadowing geometry.

In the second, final pass, the scene is rendered with conventional shading, but the contribution of a lightsource is modulated with a visibility factor obtained from its shadow map. The visibility of a shaded point is determined by projecting it onto the shadow map, and comparing it to the shadowing geometry sample there. This yields a binary visibility factor. If \vec{z} is the position of the light source, and $\delta(\vec{z} \rightarrow \vec{x})$ is the depth value read from the shadow map at direction $\vec{z} \rightarrow \vec{x}$ then the visibility factor is

$$v(\vec{x}) = \begin{cases} 1 & \text{if } \delta(\vec{z} \rightarrow \vec{x}) \geq |\vec{z} - \vec{x}| \\ 0 & \text{if } \delta(\vec{z} \rightarrow \vec{x}) < |\vec{z} - \vec{x}| \end{cases}$$

Shadow maps have a finite resolution, and sampling artifacts arise. Shadow contours will follow lexel boundaries, resulting in disturbing jagged edges, exhibiting a noise pattern at the frequency of shadow map sampling. This high-frequency noise can be filtered out by a low-pass filter. However, there is a very important caveat. It is the binary visibility factor that we want to have filtered, which depends on the depth values stored in the shadow map strongly non-linearly. Thus, on a conventional shadow map, only post-filtering – that is, filtering after point-sampling – can be applied. This is called *percentage closer filtering*^{5, 2}, as the result is the ratio of shadowing sample points compared to which the shaded surface point is closer to the lightsource. With just four sample points, bilinear filtering can be applied, which is hardware-supported on GPUs. For better quality results, more samples are needed, but making use of the *texture gather* feature and cache coherence, these techniques are feasible, robust, and therefore popular in real-time applications.

In order to make pre-filtering possible, the contents of the shadow map have to be extended and reinterpreted so that linear filtering is allowable over it. The idea that allows this is that of *variance shadow maps*¹. Instead of storing depth

values (the filtering of which is meaningless in itself), a distribution of depths is stored in every lexel. Thus, a new random variable D is introduced, and the visibility factor is defined to be the probability $\Pr(D \geq t)$ of D being larger than depth t of the shadowed point. The distribution is represented by its first two moments, that is, the mean value M_1 (which is, before filtering, identical to the original depth) and the mean squared M_2 . The variance of the distribution can be obtained from the moments as

$$\sigma^2 = M_2 - M_1^2.$$

Using Chebyshev's inequality an upper bound, can be obtained for the probability $\Pr(t \geq D)$ of a point at depth t being behind the occluding surface

$$\Pr(t \geq D) \leq p_{\max}(t) = \frac{\sigma^2}{\sigma^2 + (t - M_1)^2} \quad (4)$$

when $t \geq M_1$. This has been shown to be a close approximation for shading surface points in practice. Thus the visibility can be expressed as

$$v(\vec{x}) = \begin{cases} p_{\max}(|\vec{z} - \vec{x}|) & \text{if } \delta(\vec{z} \rightarrow \vec{x}) \geq |\vec{z} - \vec{x}| \\ 0 & \text{if } \delta(\vec{z} \rightarrow \vec{x}) < |\vec{z} - \vec{x}| \end{cases}$$

A shadow map that contains both moments can be pre-filtered with conventional texture filtering methods ranging from mipmapping to separable Gauss filtering. This will maintain the concept of the texel values representing a distribution of depth values, only over a larger area. During final rendering, a single texel has to be sampled to compute the p_{\max} value and obtain a visibility estimate.

2. Previous work

2.1. Ray marching

We can approximate the volumetric rendering equation with a ray marching method that iteratively evaluates Equation 3. For every pixel of the final image, a ray is shot from the eye, sampled at regular intervals, summing the contribution of individual samples.

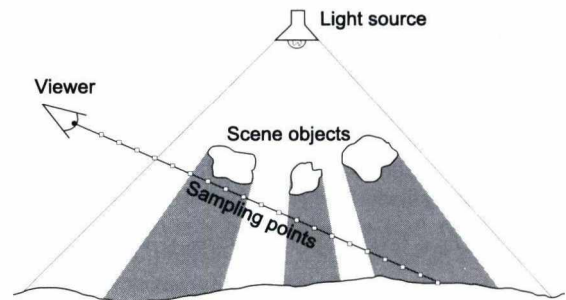


Figure 1: Ray marching.

The algorithm executes the following steps (Figure 1):

1. In every pixel it determines the visible surface point and its reflected radiance.
2. It iterates along the ray from the surface to the camera making small steps. In a particular sample point on the ray
 - the shadow map is queried for visibility factor $v(\vec{y}_n)$,
 - in-scattering term $L_{\text{light}}(\vec{y}_n, \omega_{\text{light}})P(\omega_{\text{light}}, \omega)$ is computed,
 - absorption factor $e^{-\tau n \Delta s}$ from the sample point to the eye is obtained, and
 - their product is added to the accumulated radiance.
3. The accumulated radiance is stored in the pixel spear-headed by the ray.

2.2. Interleaved sampling

Tóth and Umenhoffer have proposed an image-space filtering technique using interleaved sampling^{9,3}, which can be considered a post-processing technique from the point of view of shadow map sampling. Samples of neighboring pixels are reused, allowing for a better overall sampling with very few samples evaluated for a pixel. However, this technique blurs the edges of light shafts and it may suffer from minor artifacts near shaft edges.

3. New algorithms

For shadow maps, pre-filtering techniques, most notably variance shadow maps, have been used successfully in real-time applications. They trade expensive shading-time filtering for a relatively cheap pre-processing of the shadow map, unavoidably losing some information and accuracy.

Classic shadow map filtering techniques like percentage closer filtering and variance shadow maps are useful to eliminate aliasing of the visibility factor in volumetric rendering just like when shading surfaces. However, in Equation 2, the visibility term is integrated, effectively realizing a kind of filtering along the ray. Indeed, the cross-section contours of light shafts do not show up directly on the final rendered image, and they are mostly unimportant for visual quality. However, the integral itself is sampled when performing ray marching, the effects of which can be very much visible as a layering pattern perceivable in light shafts. Shadow map filtering also alleviates this problem to some extent, as it partly works as a filtering over the eye ray. However, a filtering technique that specifically aims to concentrate all processing power for filtering along the ray should perform better.

Our idea is based on applying pre-filtering on the shadow map that serves the purposes of ray marching. During marching, integration is carried out along rays originating from the eye. In the shadow map's image space, these rays all start from the transformed position of the eye, which may or may not be within the shadow map. Thus, pre-filtering

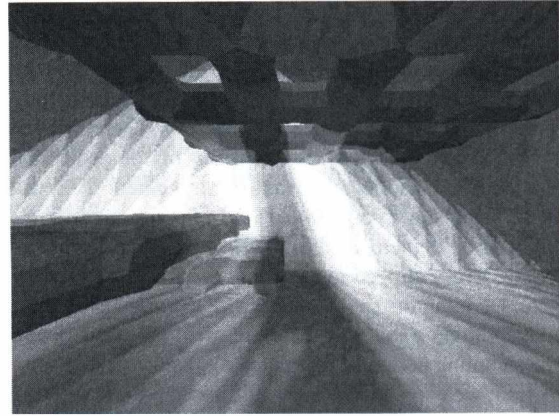


Figure 2: Layering artifact due to a large marching step.

along these rays would be accomplished using a radial filtering pass. For ease of discussion, we will refer to a half-line originating from the eye in the shadow map space as a *spoke*. There are a lot of eye rays that map to the same spoke. They are aligned on one of the planes that contain both the light source and the eye. When rendering from the eye, such a spoke plane would be visible as a line on the screen.

4. Radially filtered variance shadow map

As described in the previous section, eye rays are mapped to spokes in the shadow map, originating from the eye position transformed into shadow map space. When evaluating Equation 2 using ray marching, we need to pre-filter the integrand to reduce sampling artifacts. As integration is performed over a ray, filtering should also happen over this domain. Even though we cannot pre-process the shadow map with respect to one particular eye ray, if we filter along a spoke then it can be considered a filtering for all rays mapped to that spoke. This can be accomplished using a radial filtering step on the shadow map, which is an image processing procedure that can be efficiently performed by the graphics card. A full-viewport quadrilateral has to be rendered to a render target texture, where pixel shaders sample the input image according to a one-dimensional filter kernel along the spoke direction. In order to allow for the pre-filtering of the shadow map data, we have to use variance shadow maps, that is, we also need to store the second moments.

During ray marching, we are sampling the shadow map along the ray with a certain sampling frequency. According to the *Shannon-Nyquist sampling theorem*⁴, the sampling frequency must be the double of the highest frequency component of the sampled signal, in order to avoid aliasing. Thus, our pre-processing should ideally realize a low-pass filter cutting off frequency components above the half of the sampling frequency of ray marching.

The ray marching algorithm always uses a fixed number

of samples. According to Equation 2, the integration is performed up to ray length S . However, it would be a waste of resources to evaluate samples which are not mapped onto the shadow map, as the incoming light term, and consequently the contribution of the sample will be zero. Therefore, we can clip the ray onto the view frustum of the projection with which the shadow map has been rendered. This can be done using the *Cohen-Sutherland* clipping algorithm⁶, which must be implemented in the shader that performs ray marching. Every clipped ray segment maps to a segment of a spoke. Dividing the spoke length with the number of samples taken along a ray will therefore yield an upper bound for the sampling period of ray marching. Knowing this sample period, we can adjust the kernel size of the pre-filtering to match the Shannon-Nyquist criterion.

5. Root finding methods

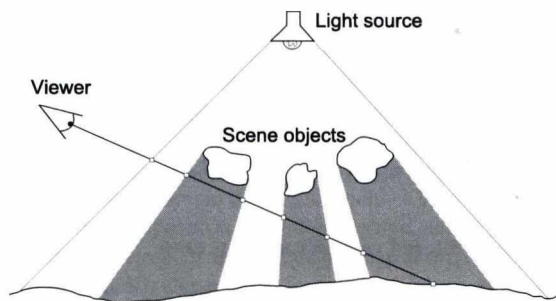


Figure 3: Shadow entry and exit points.

As a ray travels through space, it may enter and exit the shadowed volume multiple times, but the complete ray can always be separated into homogeneously lighted and shadowed segments. The information we actually need to obtain from the shadow map is where these segments start or end, that is, where the ray crosses the shadow volume boundary (see Figure 3). This means we need to find where $\delta(\vec{z} \rightarrow \vec{x}(s)) - |\vec{z} - \vec{x}(s)|$ changes sign. If $\delta(\vec{z} \rightarrow \vec{x}(s))$ is pre-filtered to get $\delta^*(\vec{z} \rightarrow \vec{x}(s))$ as described in the previous section, then δ^* can be considered a smooth continuous function, on which root finding methods may work effectively. This involves specifying a search segment, and then subdividing it recursively by taking a new sample within the segment.

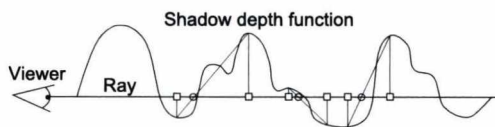


Figure 4: Subdivision with the regula falsi method.

If one of the neighboring sample points is lighted and the

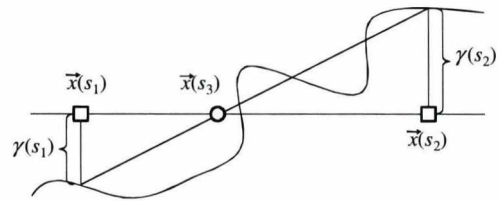


Figure 5: Regula falsi root finding step.

other one is in shadow, we can use the *regula falsi* root finding method to insert a new sample between them (see Figure 4 for a depiction of the subdivision process, and Figure 5 for the nomenclature of the subdivision formula). Let us denote the difference of the shadowing and the shadowed depth as

$$\gamma(s) = |\vec{z} - \vec{x}(s)| - \delta^*(\vec{z} \rightarrow \vec{x}(s)).$$

If the two original sample points are $\vec{x}(s_1)$ and $\vec{x}(s_2)$, the new sample point is computed as

$$\vec{x}(s_3) = \frac{\gamma(s_2)\vec{x}(s_1) + \gamma(s_1)\vec{x}(s_2)}{\gamma(s_1) + \gamma(s_2)}.$$

The subdivision can be repeated until all sample distances are below a desired quality threshold.

However, we do not only need to find one root, but all of them. Between two samples which are both shadowed or lighted, a differently illuminated segment is always possible. Classic root finding methods are useless here, as a linear interpolation yields no intersections over the segment. Intuitively, there is a higher chance of finding a sample point with lighting different from that of the endpoints where the ray is nearer to the surface, that is, where $\gamma(s)$ is smaller. However, this heuristic fails to recognize that the location of shadow boundaries is a more important factor. We can guess if a shadow boundary is nearby if we use variance shadow maps, and the second moments have also been radially filtered. In this case we can compute the variance of the shadowing depth, which can be used in the Chebyshev inequality (Equation 4). Thus we obtain an upper bound for the probability of a point in the neighborhood of a sampling point being in shadow. In case of a lighted sample point, this is also the probability of a shadow boundary being in this neighborhood. For the opposite case, we can exploit the symmetry of the situation, and extend the definition of p_{max} to be valid when $\delta(\vec{z} \rightarrow \vec{x}) < |\vec{z} - \vec{x}|$.

When subdividing the segment between the sample points, we can use the p_{max} probabilities as weights to combine the endpoint so that the new sample will be closer to the more uncertain endpoint.

Our measurements with recursive root finding algorithms implemented on the graphics card provided disappointing results, ranging from no improvement to an outright 50%

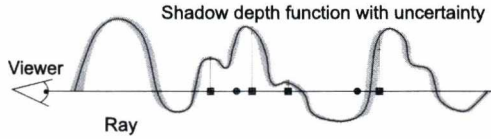


Figure 6: Subdivision step for sample points with identical visibility.

performance loss (see Section 7 for details). Finding a root typically requires at least three or four texture fetches. With the ray crossing ten shadow boundaries, that already means thirty texture fetches, comparable to what would provide acceptable quality images with straightforward ray marching of a filtered map. Thus, with this low initial number of samples, it is difficult to reduce it further using an adaptive scheme. What little can be gained is quickly outweighed by the overhead costs. Recall that straightforward ray marching is a simple, iterative algorithm. Firstly, managing a recursive algorithm in a GPU shader adds significant instruction count and storage needs. Higher storage may mean less threads running concurrently. Second, samples are taken randomly instead of reading them linearly along the ray. This is detrimental to texture cache coherence.

6. Adaptive marching

Despite the failure of root finding methods to provide a speedup for ray marching, the idea of the variance shadow map to be used to predict the local variability of the lighting can be used for an easy trick. Instead of marching the ray in uniform steps, we may take larger steps where the shadow boundary is far and depth has a low variation, and making the samples denser where the probability of crossing a shadow boundary is large. Recall that the p_{\max} value obtained from a variance shadow map was an overestimation for the probability of a surface point being visible from the light. If we extend the p_{\max} formula to be valid where the depth is less than the mean, it overestimates the probability of a surface point being in shadow. We know that the depth function is stored in a shadow map, so we can assume, without the loss of accuracy, that the ray can be divided into segments of length Δs_{\min} , within every one of which light source visibility is constant. We also err on the safe side if we suppose that the visibility of different segments is statistically independent. Value $1 - p_{\max}$ underestimates the probability of a point being in shadow. The probability of all points being in shadow over L segments is underestimated by $(1 - p_{\max})^L$. Symmetrically, if $d < M_1$, the same formula applies for all segments being visible. Thus, $(1 - p_{\max})^L$ underestimates the probability of no visibility change over L segments. If we specify that this should remain over a parameter R , for the number of allowed segments we get the formula

$$L < \frac{\log R}{\log 1 - p_{\max}},$$

meaning the step size should be

$$\Delta s = \Delta s_{\min} \frac{\log R}{\log 1 - p_{\max}}.$$

However, function $-1/\log(1 - x)$ can be approximated by a linear formula in range $[0.3, 0.99]$ within a 20% error margin, and for lower p_{\max} values we err on the safe side again by not allowing the step size to go infinitely large. Thus, a simplified linear formula for the step size can be written as

$$\Delta s = p_{\max} \Delta s_{\min} + (1 - p_{\max}) \Delta s_{\max},$$

where Δs_{\min} and Δs_{\max} are the chosen minimum and maximum step sizes. The choice of these parameters should depend on the shadow map resolution, the scene complexity, and the desired quality, just like sample number N in the original ray marching algorithm. The number of samples is not necessarily constant any more, but it remains between $S/\Delta s_{\max}$ and $S/\Delta s_{\min}$.

Thus, with a radially filtered variance shadow map, and very little modification of the ray marching algorithm, it is possible to decrease the number of required samples significantly.

7. Results

We made measurements in two test scenes. The first is the Buddha scene (Figure 7) with a polygon count of 160K, but a fairly simple shadow pattern with a single occluding object. The second is the Fan-in-room scene (Figure 8), where the triangle throughput was not a limiting factor, but shadow casters had high depth complexity. Images were rendered in 640×480 resolution on an NVIDIA GeForce GTX 260 graphics card. The algorithm parameters were tuned to work without observable artifacts, thus they all rendered visually indistinguishable images. As the frame rate depends on the position of the camera, we conducted several experiments. Average frame rates (frames per second) can be summarized as:

	Buddha scene	Fan-in-room scene
Ray marching	99	144
Root finding	48	147
Adaptive steps	162	458

The difference between the performance of ray marching is due to the different polygon counts of the two scenes, it is in fact the case that the second scenario required more ray marching samples. In the Buddha scene, the sampling algorithm based on root finding heuristics exhibited a dismal performance, where there were very few shadow boundaries to find, so the resulting sampling was just a uniform sampling created in a convoluted way, and the overhead of the algorithm did not pay off. Ray marching with adaptive steps, on the other hand, could reduce the required sample

count, and achieved a speedup of 60%. In the Fan-in-room scene, the depth pattern was more complex, and the root finding heuristics could make up for its drawbacks, but it could still not outperform straightforward ray marching. The adaptive steps algorithm, on the other hand, could render three times as many frames in a second as the original ray marching method, as it could distribute the samples exactly where they were needed.

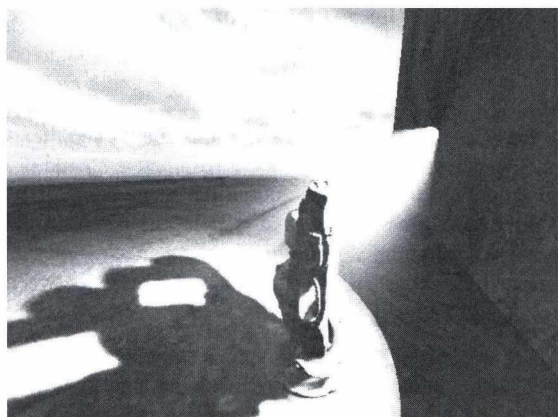


Figure 7: Sun rays in the buddha scene.

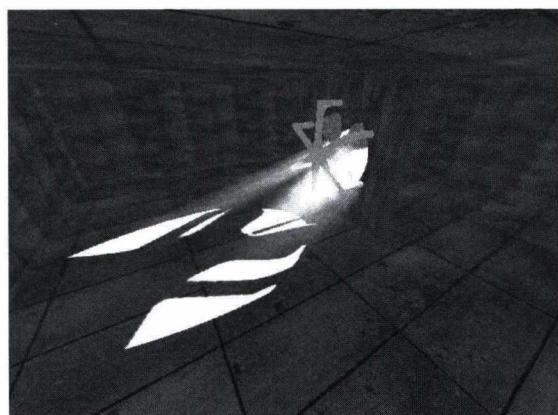


Figure 8: Light shaft in the fan-in-room scene.

8. Conclusion

Shadow maps, when used for finding the visibility factor in the radiative transport equation for rendering participating media, should be filtered differently. Filtering along eye rays corresponds to a radial blur operation in the shadow map space, which can be performed over a variance shadow map in a preprocessing step. Extra information in this map can then be used to distribute samples more optimally, resulting in a reduced sample count and increased rendering performance. However, the adaptive heuristic of this sample placement must be lightweight enough to match the brute force

simplicity of the ray marching algorithm and avoid the overhead that might negate or reverse the gains.

It is left to future work to investigate other shadow map pre-filtering techniques, for instance minimum-mean-maximum filtering, that might allow for an even better prediction of shadow boundaries as variance shadow maps. We also plan to improve the radial processing idea into a method that eliminates the approximation of the ray marching summation and discards the filtering approach in favor of an exact, geometric one.

9. Acknowledgement

This work has been supported by the Teratomo project of the National Office of Research and Technology, OTKA K-719922 (Hungary), and the Hungarian-Croatian Fund.

References

1. W. Donnelly and A. Lauritzen. Variance shadow maps. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, page 165. ACM, 2006. 3
2. R. Fernando. Percentage-closer soft shadows. In *ACM SIGGRAPH 2005 Sketches*, page 35. ACM, 2005. 3
3. A. Keller and W. Heidrich. Interleaved sampling. In *Rendering Techniques 2001: Proceedings of the Eurographics Workshop in London, United Kingdom, June 25-27, 2001*, page 269. Springer Verlag Wien, 2001. 4
4. H.D. Lueke. The origins of the sampling theorem. *IEEE Communications Magazine*, 37(4):106–108, 1999. 4
5. W.T. Reeves, D.H. Salesin, and R.L. Cook. Rendering antialiased shadows with depth maps. *ACM SIGGRAPH Computer Graphics*, 21(4):291, 1987. 3
6. R.F. Sproull and W.M. Newman. *Principles of interactive computer graphics*. McGraw-Hill, Inc. New York, NY, USA, 1979. 5
7. L. Szirmay-Kalos. Monte-Carlo Methods in Global Illumination. *Photo-realistic Rendering with Randomization*. VDM, Verlag Dr. Müller, Saarbrücken, 2008. 2
8. L. Szirmay-Kalos, L. Szécsi, and M. Sbert. GPU-Based Techniques for Global Illumination Effects. *Synthesis Lectures on Computer Graphics and Animation*, 2(1):1–275, 2008. 2
9. B. Tóth and T. Umenhoffer. Real-time Volumetric Lighting in Participating Media. In *Eurographics Short Papers*, 2009. 4
10. L. Williams. Casting curved shadows on curved surfaces. *ACM SIGGRAPH Computer Graphics*, 12(3):270–274, 1978. 3

Adaptive Sampling with Error Control

László Szécsi¹, László Szirmay-Kalos¹, and Murat Kurt²

¹: BME IIT, Hungary, ²: International Computer Institute, Ege University, Turkey

Abstract

This paper proposes a multi-dimensional adaptive sampling algorithm for rendering applications. Unlike importance sampling, adaptive sampling does not try to mimic the integrand with analytically integrable and invertible densities, but approximates the integrand with analytically integrable functions, thus it is more appropriate for complex integrands, which correspond to difficult lighting conditions and sophisticated BRDF models. We develop an adaptation scheme that is based on ray-differentials and does not require neighbor finding and complex data structures in higher dimensions. As a side result of the adaptation criterion, the algorithm also provides error estimates, which are essential in predictive rendering applications.

1. Introduction

Rendering is mathematically equivalent to the evaluation of high-dimensional integrals. In order to avoid the curse of dimensionality, Monte Carlo or quasi-Monte Carlo quadratures are applied, which take discrete samples and approximate the integral as the weighted sum of the integrand values in these samples. The challenge of these methods is to find a sampling strategy that generates a small number of samples providing an accurate quadrature.

Importance sampling would mimic the integrand with the density of the samples. However, finding a proper density function and generating samples with its distribution are non-trivial tasks. There are two fundamentally different approaches to generating samples with a probability density. The *inversion method* maps uniformly distributed samples with the inverse of the cumulative probability distribution of the desired density. However, this requires that the density is integrable and its integral is analytically invertible. The integrand of the rendering equation is a product of BRDFs and cosine factors of multiple reflections, and of the incident radiance at the end of a path. Due to the imposed requirements, importance sampling can take into account only a single factor of this product form integral, and even the single factor is just approximately mimicked except for some very simple BRDF models. *Rejection sampling* based techniques, on the other hand, do not impose such requirements on the density. However, rejection sampling may ignore many, already generated samples, which may lead to unpredictable

performance degradation. Rejection sampling inspired many sampling methods^{5, 28, 1, 20, 22, 27}. We note that in the context of Metropolis sampling there have been proposals to exploit even the rejected samples having re-weighted them^{30, 10, 17, 13}. *Hierarchical sampling* strategies attack product form integrands by mimicking just one factor initially, then improving the sample distribution in the second step either by making the sampling distribution more proportional to the integrand^{5, 28, 6, 7, 22} or by making the empirical distribution of the samples closer to the continuous sample distribution^{1, 20, 27}.

An alternative method of finding samples for integral quadratures is *adaptive sampling* that uses the samples to define an approximation of the integrand, which is then analytically integrated. Adaptation is guided by the statistical analysis of the samples in the sub-domains. Should the variance of the integrand in a sub-domain be high, the sub-domain is broken down to smaller sub-domains. Adaptive sampling does not pay attention to the placement of the samples in the sub-domains. However, placing samples in a sub-domain without considering the distribution of samples in neighboring sub-domains will result in very uneven distribution in higher dimensional spaces. On the other hand, variance calculation needs neighbors finding, which gets also more difficult in higher dimensions. Thus, adaptive sampling usually suffers from the curse of dimensionality.

The *VEGAS method*¹⁸ is an adaptive Monte-Carlo technique that generates a probability density for importance

sampling automatically and in a separable form. The reason of the requirement of separability is that probability densities are computed from and stored in histogram tables and s number of 1-dimensional tables need much less storage space than a single s -dimensional table.

This paper proposes a multi-dimensional adaptive sampling scheme, which:

- works with vector valued functions as well and does not require the introduction of simple scalar valued importance unlike in importance sampling algorithms,
- is simple to implement even in higher dimensions,
- has small additional overhead both in terms of storage and computation, because it does not require complex data structures or neighborhood searches as other adaptive algorithms,
- provides consistent, deterministic estimates with small and predictable quadrature error (the term unbiasedness is not applicable for deterministic sampling algorithms).

2. Previous work

Adaptive sampling: Adaptive sampling uses samples to approximate the integrand, thus concentrates samples where the integrand changes significantly. The targeted approximation is usually piece-wise constant or piece-wise linear¹⁴. Image space adaptive sampling was used even in the first ray tracer³¹, and has been improved by randomization⁴ and by the application of information theory tools²¹. Adaptive sampling methods should decompose the sampling domain, which is straightforward in 2D but needs special data structures like the kd-tree in higher dimensions¹¹.

Ray- and path-differentials: A ray-differential means the derivative of the contribution of a path with respect to some variable defining this path. Igehy introduced ray-differentials with respect to screen-space coordinates¹². Ray-differentials were used in filtering applications where the differentials define the footprint of the samples, i.e. the domain the filter. Suykens²⁵ generalized this idea for arbitrary sampling parameters defining the path. *Photon differentials*²⁴ also use this concept to enhance radiance estimates in photon mapping. We note that unlike these techniques, we do not use path-differentials in this paper for setting up filters, but to guide the sampling process.

Error estimation in rendering: In *predictive rendering* we need not only an estimate of the image, but also error bounds describing the accuracy of the method²⁹. Unfortunately, the computation of the error of a rendering algorithm is even more complex than producing the image². Error analysis was proposed in the context of finite-element based radiosity methods^{19, 3, 4, 23}. For Monte-Carlo methods, we can use the statistics of the samples to estimate the variance of the estimator. For deterministic approaches, in theory, the Koksma-Hlawka inequality could be used. However, due to the in-

finite variation of the typical integrands in rendering, this provides useful bounds only in exceptional cases²⁶.

3. Proposed sampling scheme

For the sake of notational simplicity, let us consider first a 1D integral of a possibly vector valued integrand $f(t)$ in the domain of $[0, 1]$. Suppose that the integration domain is decomposed to intervals $\Delta^{(1)}, \Delta^{(2)}, \dots, \Delta^{(M)}$ and one sample point $t^{(i)}$ is selected in each of them. The integral is estimated in the following way:

$$\int_0^1 f(t) dt \approx \sum_{i=1}^M f(t^{(i)}) \Delta^{(i)}. \quad (1)$$

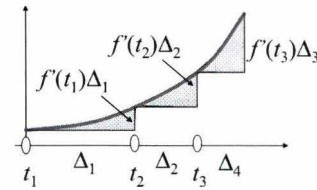


Figure 1: The error of adaptive sampling is the total area of triangles of bases Δ_i and heights that are proportional to Δ_i and to the derivative of integrand f .

Looking at Figure 1, we can conclude that the error of the integral is the sum of small triangles in between the integrand and its piece-wise constant approximation, which can be expressed as follows if derivative f' exists:

$$\left| \int_0^1 f(t) dt - \sum_{i=1}^M f(t^{(i)}) \Delta^{(i)} \right| \approx \sum_{i=1}^M |f'(t^{(i)})| \frac{(\Delta^{(i)})^2}{2}. \quad (2)$$

We minimize the error with the constraint $\sum_{i=1}^M \Delta^{(i)} = 1$ using the Lagrange multiplier method. The target function that also includes the constraint is

$$\sum_{i=1}^M |f'(t^{(i)})| \frac{(\Delta^{(i)})^2}{2} - \lambda \left(\sum \Delta^{(i)} - 1 \right).$$

Computing the partial derivatives with respect to $\Delta^{(i)}$ and making it equal to zero, we obtain:

$$|f'(t^{(i)})| \Delta^{(i)} = \lambda, \quad i = 1, 2, \dots, M,$$

thus, the variation in all sub-domains should be equal, i.e. the sampling scheme should concentrate samples where the integrand changes quickly.

3.1. Extension to integrands typical in rendering

The integrand of rendering is defined in a high-dimensional path space, which is transformed to a D -dimensional unit cube \mathcal{U} . A point in the unit cube is denoted by $\mathbf{u} =$

(u_1, \dots, u_D) . The integrand is high-dimensional, can only be point sampled, but together with point sampling, its derivative can also be obtained according to the concept of *ray-differentials*.

Adaptive sampling should decompose the integration domain, i.e. the unit cube to M sub-domains. Sub-domain i has edge lengths $\Delta_1^{(i)}, \dots, \Delta_D^{(i)}$ and volume

$$\Delta^{(i)} = \Delta_1^{(i)} \cdot \dots \cdot \Delta_D^{(i)}.$$

Each sub-domain contains a single sample $\mathbf{u}^{(i)}$ from which the integral is estimated as:

$$\int_{\mathcal{U}} F(\mathbf{u}) d\mathbf{u} \approx \sum_{i=1}^M F(\mathbf{u}^{(i)}) \Delta^{(i)}. \quad (3)$$

In order to find the error of this multi-dimensional integral, we approximate the integrand by the first-order Taylor series in each sub-domain:

$$F(\mathbf{u}) \approx F(\mathbf{u}_i) + \sum_{d=1}^D \frac{\partial F(\mathbf{u}^{(i)})}{\partial u_d} (u_d - u_d^{(i)}).$$

Substituting this approximation into the quadrature error, we get:

$$\left| \int_{\mathcal{U}} F(\mathbf{u}) d\mathbf{u} - \sum_{i=1}^M F(\mathbf{u}^{(i)}) \Delta^{(i)} \right| \approx \sum_{i=1}^M \sum_{d=1}^D \left| \frac{\partial F(\mathbf{u}^{(i)})}{\partial u_d} \right| \frac{\Delta_d^{(i)}}{2} \Delta^{(i)}. \quad (4)$$

We minimize this error by finding the sub-domain sizes $\Delta_d^{(i)}$ satisfying the constraints that sub-domains are disjunct and their union is the original unit cube.

If sub-domain $\Delta^{(i)}$ is subdivided along axis d , the error is reduced proportionally by

$$\epsilon_d^{(i)} = \left| \frac{\partial F(\mathbf{u}^{(i)})}{\partial u_d} \right| \Delta_d^{(i)} \Delta^{(i)}.$$

To maximize the error reduction, that sub-domain and dimension should be selected for subdivision where this product is maximum. As each sub-domain stores a single sample, the reduced error can also be assigned to the samples. Samples are generated one-by-one. When we are at sample \mathbf{u}_i , the integrand as well its partial derivatives are computed, and each sample is given the vector of possible error reductions:

$$\epsilon^{(i)} = [\epsilon_1^{(i)}, \dots, \epsilon_D^{(i)}].$$

The maximum error reduction with respect to all samples and dimensions tells us which sub-domain should be subdivided and also specifies the axis of the subdivision. The sub-domain is subdivided generating new samples in the neighborhood of the sample associated with the subdivided cell.

3.2. Sample generation and sub-domain subdivision

In order to explore the integration domain and establish the subdivision scheme, we take a low-discrepancy series ^{16,9}.

We work with the Halton series, but other low discrepancy series, such as the Sobol series or (t, m, s) -nets with $t = 0$ could be used as well ¹⁵. For the 1D Halton sequence in base B , the elemental interval property means that the sequence will visit all intervals of length $[kB^{-R}, (k+1)B^{-R}]$ before putting a new point in an interval already visited. As the sequence is generated, the algorithm produces a single point in each interval of length B^{-1} , then in each interval of length B^{-2} , etc.

For a D -dimensional sequence of relative prime bases B_1, \dots, B_D , this property means that the sequence will insert a sample in each cell

$$[k_1 B_1^{-R_1}, (k_1 + 1) B_1^{-R_1}] \times \dots \times [k_D B_D^{-R_D}, (k_D + 1) B_D^{-R_D}]$$

before placing a second sample in any of these cells. The elemental interval property thus implicitly defines an automatically refined sub-domain structure, which can also be exploited in adaptive sampling.

If we decide to subdivide the sub-domain of sample i along coordinate axis d , then the following new samples must be generated:

$$i + \prod_{d=1}^D B_d^{R_d}, \quad i + 2 \prod_{d=1}^D B_d^{R_d}, \quad \dots, \quad (B_d - 1) \prod_{d=1}^D B_d^{R_d}.$$

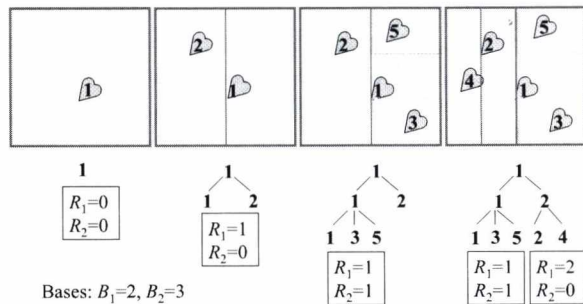


Figure 2: Evolution of samples and their corresponding sub-domains in 2D with a Halton sequence of bases $B_1 = 2$ and $B_2 = 3$.

In Figure 2 we show the evolution of the cell structure in 2D as new samples are introduced. First, we have a single sample $j = 1$ in a cell of volume 1 ($R_1 = 0, R_2 = 0$). Then, this cell is subdivided along the first axis where the basis is $B_1 = 2$, generating a new sample $j = 1 + B_1^{-R_1} B_2^{-R_2} = 2$. Simultaneously, the size of the cell is reduced to half ($R_1 = 1, R_2 = 0$). In the third step, the cell of sample $j = 1$ is subdivided again, but now along the second axis. This happens by producing new samples $j = 1 + B_1^{-R_1} B_2^{-R_2} = 3$ and $j = 1 + 2B_1^{-R_1} B_2^{-R_2} = 5$, with cell sizes as ($R_1 = 1, R_2 = 1$). Finally the cell of sample $j = 2$ is subdivided into two parts.

The sampling process can be visualized as a tree (Figure 2) where upper level nodes are the subdivided sub-domains, and leaves are the cells containing exactly one

sample. The number of children of a node equals to the corresponding prime basis number of the Halton sequence. Although only leaves are relevant for the approximation, it is worth maintaining the complete tree structure since it also helps to find that leaf which should be subdivided to reduce the error most effectively. In upper level nodes, we maintain the maximum error of its children.

The generation of a new sample is the traversal of this tree. When traversing a node, we continue descending into the direction of the child with maximum error. When we arrive at a leaf, the dimension which reduces the error the most is subdivided and the leaf becomes a node with children including the original sample and the new samples. While ascending on the tree, the maximum error of the new nodes are propagated.

The recursive function implementing this scheme gets a node c . Calling this function with the root node, it generates a new sample that reduces the integration error the most, and updates the complete tree as well:

```

Process(c)
  if (c is a leaf)
    i = sample id of this leaf
    (R1, ..., RD) = subdivision levels of this node
    (ε1, ..., εD) = errors of this node
    Find the dimension  $d$  of the highest error ε $d$ 
    for k = 1 to B $d$  - 1 do
      Generate sample with id  $j = i + k \prod_{d=1}^D B_d^{R_d}$ 
      Compute  $F(\mathbf{u}_j)$  and the partial derivatives
      Add sample  $j$  as child to node  $c$ 
    endfor
    Update subdivision levels in sample  $i$ 
    Update error in sample  $i$ 
    Add sample  $i$  as child to node  $c$ 
    ε(c) = maximum error of the children
  else
    Find the child node  $c_{\text{first}}$  with maximum error
    Find the second largest error εsecond
    εc = max( Process( $c_{\text{first}}$ ), εsecond )
  endif
  return ε(c)
end

```

This algorithm works in arbitrary dimensions. In order to use the method, first we have to map the integration problem to the unit cube, and have to establish the formulae for computing the derivatives. This seems complicated, but in fact, it is quite simple. When we have the program of evaluating the integrand, the derivatives can be obtained by simply deriving the program code. In the following section, we discuss a simple application of the method for environment mapping.

3.3. Robustness issues

So far we assumed that the derivative exists and is an appropriate measure for the change of the function in a sub-domain. This is untrue where the function has a discontinuity

or when the sub-domain is large. From another point of view, a *consistent estimator* should decompose a sub-domain even if the derivative is zero at the examined sample when the number of samples goes to infinity. To obtain this behavior, a positive constant is added to the derivative before the error is computed. This constant is reduced as size of the sub-domain gets smaller, corresponding to the fact that for smaller sub-domains the derivative becomes a more reliable measure of the change of a function.

Note also that the derivative does not exist where the integrand is discontinuous. This is typically the case when the integrand includes the visibility factor. Ignoring the visibility factor in the derivative computation, we can still apply the method since we use the derivatives only to decide the "optimal" decomposition of the integration domain, and not directly for the estimation of the integral. If the decomposition is not "optimal" due to the ignored factors, the method will still converge to the real value as more samples are introduced, but the convergence will be slower. In theory, it would be possible to replace the product of the derivative by more general estimates of the integration error, but they also have added cost.

The proposed method uses the Halton sequence which can fill D -dimensional spaces uniformly with $O(\log^D N/N)$ discrepancy. However, we cannot claim that our method remains similarly stable in very high dimensions. The problem is the index generation needed to find additional samples in the neighborhood of a given sample, can result in fast growing sequences¹⁵. So we propose application fields where the dimension of the integration domain is moderate.

4. Application to environment mapping

Environment mapping⁸ computes the reflected radiance of point \vec{x} as

$$L(\vec{x}, \vec{\omega}) = \int_{\Omega} L^{\text{env}}(\vec{\omega}') f_r(\vec{\omega}', \vec{x}, \vec{\omega}) \cos \theta'_x v(\vec{x}, \vec{\omega}') d\omega',$$

where Ω is the set of all directions, $L^{\text{env}}(\vec{\omega}')$ is the radiance of the environment map at *illumination direction* $\vec{\omega}'$, f_r is the BRDF, θ'_x is the angle between the illumination direction and the surface normal at shaded point \vec{x} , and $v(\vec{x}, \vec{\omega}')$ is the indicator function checking whether no virtual object is seen from \vec{x} at direction $\vec{\omega}'$, so environment lighting can take effect. We suppose that the BRDF and the environment lighting are available in analytic forms.

In order to transform the integration domain, i.e. the set of illumination directions, to a unit square $u_1, u_2 \in [0, 1]$, we find a parametrization $\vec{\omega}'(u_1, u_2)$. We shall consider different options for this parametrization, including a global parametrization that is independent of the surface normal of the shaded point, and BRDF based parametrizations where the Jacobi determinant of the mapping compensates the variation of the BRDF and the cosine factor.

The integral of the reflected radiance can also be evaluated in parameter space:

$$L(\vec{x}, \vec{\omega}) = \int_0^1 \int_0^1 L^{\text{env}}(\vec{\omega}'(u_1, u_2)) R(u_1, u_2) v(\vec{x}, \vec{\omega}'(u_1, u_2)) du_1 du_2.$$

where

$$R(u_1, u_2) = f_r(\vec{\omega}'(u_1, u_2), \vec{x}, \vec{\omega}) \cos \theta_{\vec{x}}'(u_1, u_2) \frac{\partial \omega}{\partial u_1 \partial u_2}$$

is the *reflection factor* defined by the product of the BRDF, the cosine of the angle between the surface normal and the incident direction, and also the Jacobi determinant of the parametrization.

The derivatives of the integrand $F = L^{\text{env}} \cdot R \cdot v$ of environment mapping are ($d = 1, 2$):

$$\frac{\partial F}{\partial u_d} = \frac{\partial L^{\text{env}}}{\partial u_d} R v + L^{\text{env}} \frac{\partial R}{\partial u_d} v.$$

Thus, we need to evaluate the derivative of the reflection factor and of environment lighting. The derivative computation is presented in the appendix. The visibility factor is piecewise constant, so its derivative would be zero.

5. Results

The environment mapping algorithm has been implemented using the Direct3D 9 framework and run on an nVidia GeForce 8800 GFX GPU.

In environment mapping, adaptive sampling generates points in a unit square according to the product form integrand

$$L^{\text{env}}(\vec{\omega}'(u_1, u_2)) f_r(\vec{\omega}'(u_1, u_2), \vec{x}, \vec{\omega}) \cos \theta_{\vec{x}}'(u_1, u_2) \frac{\partial \omega}{\partial u_1 \partial u_2}$$

that includes the environment illumination, the cosine weighted BRDF and the Jacobi determinant of the mapping.



Figure 3: Samples generated with BRDF parametrization displayed over the transformed integrand which simplifies to $L^{\text{env}}(\vec{\omega}'(u_1, u_2))$.

In Figure 3 we depicted the samples assigned to a single shaded point. Samples and the transformed integrand are shown in parameter space. Note that the proposed adaptive sampling scheme places samples at regions where the illumination changes quickly and also takes care of the stratification of the samples.

In Figure 4 we compare the adaptive sampling scheme involving global parametrization and diffuse BRDF parametrization to classical BRDF sampling. The proposed adaptive scheme has significantly reduced the noise. BRDF parametrization is better than global parametrization as it automatically ignores that part of the integration domain where the cosine term would be zero.

Finally in the bottom row of Figure 4 we demonstrate the adaptive sampling approach in specular surface rendering. This figure compares classical BRDF sampling, adaptive sampling with global parametrization, and adaptive sampling with Phong parametrization.

6. Conclusions

This paper proposed an adaptive sampling scheme based on the elemental interval property of low-discrepancy series and the derivatives of the integrand at the samples. The derivatives provide additional information to the sampling process, which can place samples where they are really needed. The low-discrepancy series not only distributes samples globally, but it also helps refining the sample structure locally without neighborhood searches.

Having generated the samples and evaluated the integral, we have all information needed to evaluate equation (4), which provides error bounds for the estimate. Such bounds are badly needed in predictive rendering applications.

Acknowledgement

This work has been supported by the National Office for Research and Technology and OTKA K-719922 (Hungary).

Appendix: Derivative computation

6.1. Global parametrization

In the case of *global parametrization* the unit square is mapped to the set of directions independently of the local orientation of the surface. The incident direction is obtained in spherical coordinates $\phi = 2\pi u_1$, $\theta = \pi u_2$, which are converted to Cartesian coordinates of the system defined by basis vectors $\vec{i}, \vec{j}, \vec{k}$:

$$\vec{\omega}'(u_1, u_2) =$$

$$\vec{i} \cos(2\pi u_1) \sin(\pi u_2) + \vec{j} \sin(2\pi u_1) \sin(\pi u_2) + \vec{k} \cos(\pi u_2).$$

The derivatives of the incident direction are as follows:

$$\frac{\partial \vec{\omega}'}{\partial u_1} = -2\pi \vec{i} \sin(2\pi u_1) \sin(\pi u_2) + 2\pi \vec{j} \cos(2\pi u_1) \sin(\pi u_2),$$

$$\frac{\partial \vec{\omega}'}{\partial u_2} = \pi \vec{i} \sin(2\pi u_1) \cos(\pi u_2) + \pi \vec{j} \cos(2\pi u_1) \cos(\pi u_2) - \pi \vec{k} \sin(\pi u_2). \quad (5)$$

The Jacobi determinant of the global parametrization is

$$\frac{\partial \omega}{\partial u_1 \partial u_2} = 2\pi^2 \sin \theta = 2\pi^2 \sin(\pi u_2).$$

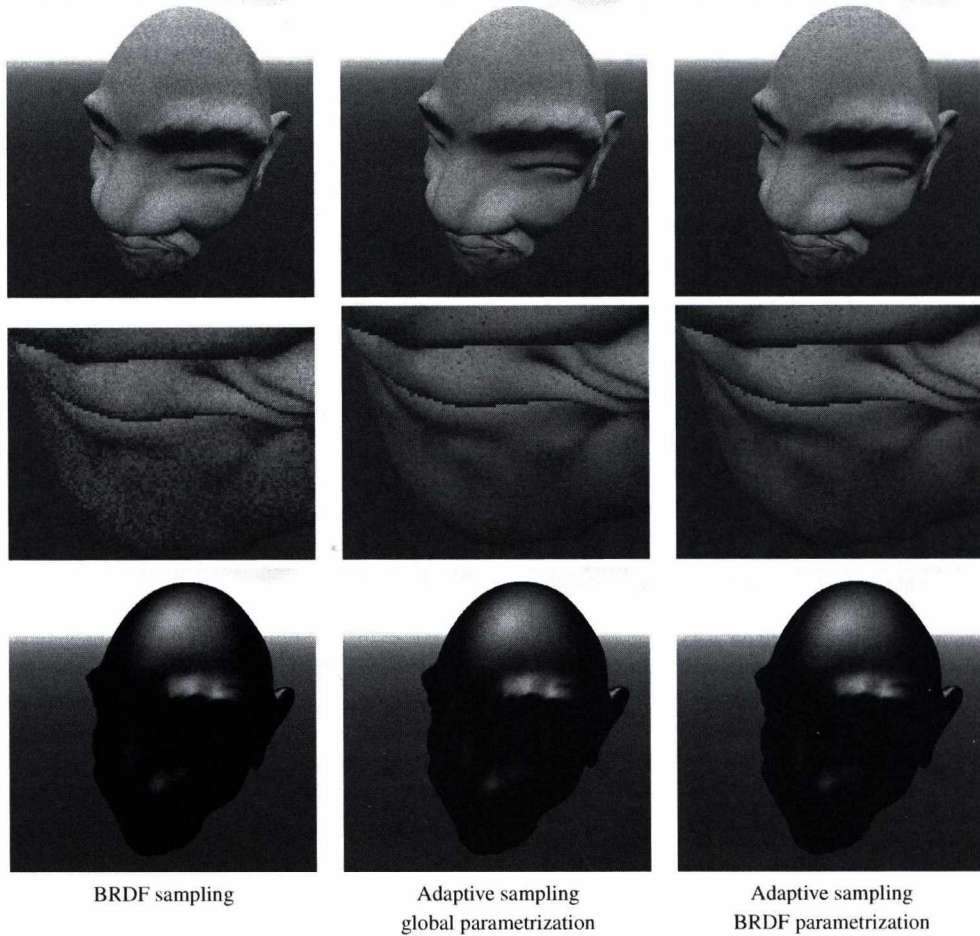


Figure 4: Diffuse objects (upper and middle row) and specular objects (bottom row) rendered with BRDF sampling, adaptive sampling with global parametrization, and adaptive sampling with diffuse BRDF parametrization.

6.2. Diffuse BRDF parametrization

We can also use the parametrization developed for BRDF sampling. The illumination direction is generated in the *tangent space* of shaded point \bar{x} , defined by surface normal $\vec{N}_{\bar{x}}$, tangent $\vec{T}_{\bar{x}}$ and binormal $\vec{B}_{\bar{x}}$, and it should mimic the cosine distribution:

$$\vec{\omega}' = \vec{T}_{\bar{x}} \cos(2\pi u_1) \sqrt{u_2} + \vec{B}_{\bar{x}} \sin(2\pi u_1) \sqrt{u_2} + \vec{N}_{\bar{x}} \sqrt{1 - u_2}.$$

The derivatives are:

$$\begin{aligned} \frac{\partial \vec{\omega}'}{\partial u_1} &= -2\pi \vec{T}_{\bar{x}} \sin(2\pi u_1) \sqrt{u_2} + 2\pi \vec{B}_{\bar{x}} \cos(2\pi u_1) \sqrt{u_2}, \\ \frac{\partial \vec{\omega}'}{\partial u_2} &= \vec{T}_{\bar{x}} \frac{\cos(2\pi u_1)}{2\sqrt{u_2}} + \vec{B}_{\bar{x}} \frac{\sin(2\pi u_1)}{2\sqrt{u_2}} - \vec{N}_{\bar{x}} \frac{1}{2\sqrt{1 - u_2}}. \end{aligned} \quad (6)$$

The Jacobi determinant compensates the cosine term:

$$\frac{\partial \omega}{\partial u_1 \partial u_2} = \frac{\pi}{\cos \theta_{\bar{x}}} = \frac{\pi}{\sqrt{1 - u_2}}.$$

6.3. Specular BRDF parametrization

In case of a Phong BRDF, the direction is generated in *reflection space* defined by the reflection direction

$$\vec{\omega}_r = 2\vec{N}_{\bar{x}}(\vec{N}_{\bar{x}} \cdot \vec{\omega}) - \vec{\omega}$$

and two other orthonormal vectors \vec{T}_r and \vec{B}_r that are perpendicular to $\vec{\omega}_r$. The mapping should follow the $(\vec{\omega}_r \cdot \vec{\omega}')^n$ function:

$$\begin{aligned} \vec{\omega}'(u_1, u_2) &= \vec{T}_r \cos(2\pi u_1) \sqrt{1 - (1 - u_2)^{\frac{2}{n+1}}} + \\ &\vec{B}_r \sin(2\pi u_1) \sqrt{1 - (1 - u_2)^{\frac{2}{n+1}}} + \vec{\omega}_r (1 - u_2)^{\frac{1}{n+1}}. \end{aligned}$$

The derivatives are obtained as:

$$\begin{aligned} \frac{\partial \vec{\omega}'}{\partial u_1} &= -2\pi \vec{T}_r \sin(2\pi u_1) \sqrt{1 - (1 - u_2)^{\frac{2}{n+1}}} + \\ &2\pi \vec{B}_r \cos(2\pi u_1) \sqrt{1 - (1 - u_2)^{\frac{2}{n+1}}}. \end{aligned}$$

$$\frac{\partial \vec{\omega}'}{\partial u_2} = \vec{T}_r \cos(2\pi u_1) \frac{u_2^{\frac{n-1}{n+1}}}{(n+1)\sqrt{1-(1-u_2)^{\frac{2}{n+1}}}} + \vec{B}_r \sin(2\pi u_1) \frac{u_2^{\frac{n-1}{n+1}}}{(n+1)\sqrt{1-(1-u_2)^{\frac{2}{n+1}}}} - \vec{\omega}_r \frac{1}{(n+1)(1-u_2)^{\frac{n}{n+1}}}. \quad (7)$$

The Jacobi determinant of the mapping is

$$\frac{\partial \omega}{\partial u_1 \partial u_2} = \frac{2\pi}{(n+1)(\vec{\omega}' \cdot \vec{\omega}')^n} = \frac{2\pi}{(n+1)(1-u_2)^{\frac{n}{n+1}}}.$$

6.4. Derivatives of the reflection factor

The reflection factor includes the Jacobi determinant of the mapping, so it must be discussed taking into account the parametrization. If we use diffuse BRDF parametrization for a diffuse surface, the Jacobi determinant will compensate the cosine term, making the reflection factor constant, and consequently its derivative equal to zero. Similarly, the specular BRDF parametrization of a specular surface also results in a zero derivative. For more complex BRDFs that have no exact importance sampling, we can still use BRDF parametrization of a similar BRDF, for example, that of the diffuse or the Phong solutions. Of course, their Jacobi determinants will not fully eliminate the BRDF and the cosine term, so we have some non-constant function that needs to be derived.

Let us now consider global parametrization, where the Jacobi determinant is $2\pi^2 \sin \pi u_2$. If the material is diffuse with diffuse reflectivity k_{dif} , and the surface normal at the shaded point \vec{x} is $\vec{N}_{\vec{x}}$, the partial derivatives of the reflection factor including the BRDF, the cosine term, and the Jacobi determinant, are ($d = 1, 2$):

$$\frac{\partial R}{\partial u_1} = 2\pi^2 k_{\text{dif}} \left(\frac{\partial \vec{\omega}'}{\partial u_1} \cdot \vec{N}_{\vec{x}} \right) \sin(\pi u_2).$$

$$\frac{\partial R}{\partial u_2} = 2\pi^2 k_{\text{dif}} \left(\left(\frac{\partial \vec{\omega}'}{\partial u_2} \cdot \vec{N}_{\vec{x}} \right) \sin(\pi u_2) + \pi (\vec{\omega}' \cdot \vec{N}_{\vec{x}}) \cos(\pi u_2) \right).$$

if $\vec{\omega}' \cdot \vec{N}_{\vec{x}} \geq 0$ and zero otherwise. The derivative of the incident direction is given in equation (5).

The derivatives can also be computed for specular BRDFs, for example, for the Phong BRDF:

$$R = 2\pi^2 k_{\text{spec}} (\vec{\omega}' \cdot \vec{\omega}_r)^n \sin(\pi u_2)$$

where k_{spec} is the specular reflectivity. Applying the rules of derivation systematically, we get:

$$\frac{\partial R}{\partial u_1} = 2\pi^2 n k_{\text{spec}} (\vec{\omega}' \cdot \vec{\omega}_r)^{n-1} \left(\frac{\partial \vec{\omega}'}{\partial u_1} \cdot \vec{\omega}_r \right) \sin(\pi u_2).$$

$$\frac{\partial R}{\partial u_2} = 2\pi^2 n k_{\text{spec}} (\vec{\omega}' \cdot \vec{\omega}_r)^{n-1} \left(\frac{\partial \vec{\omega}'}{\partial u_2} \cdot \vec{\omega}_r \right) \sin(\pi u_2) +$$

$$2\pi^3 k_{\text{spec}} (\vec{\omega}' \cdot \vec{\omega}_r)^n \cos(\pi u_2).$$

6.5. Derivatives of the environment lighting

The environment lighting may be defined analytically or by a texture map.

6.5.1. Simple sky illumination

For example, a sky illumination with the Sun at direction $\vec{\omega}_{\text{sun}}$ can be expressed as

$$L^{\text{env}}(\vec{\omega}') = L_{\text{sky}} + L_{\text{sun}}(\vec{\omega}' \cdot \vec{\omega}_{\text{sun}})^s$$

where L_{sky} and L_{sun} are the sky and sun radiances, respectively, and s is the exponent describing the directional fall-off of the sun radiance. With this environment radiance, the derivatives of the environment lighting are ($d = 1, 2$):

$$\frac{\partial L^{\text{env}}}{\partial u_d} = s L_{\text{sun}} (\vec{\omega}' \cdot \vec{\omega}_{\text{sun}})^{s-1} \left(\frac{\partial \vec{\omega}'}{\partial u_d} \cdot \vec{\omega}_{\text{sun}} \right).$$

6.5.2. Texture based environment lighting

If the environment lighting is defined by a texture map, we first project it into a spherical harmonics basis:

$$L^{\text{env}}(\vec{\omega}') = \sum_l L_{l0}^{\text{env}} Y_l^0(\cos \theta) +$$

$$\sum_l \sum_{m=-1}^{-l} L_{lm}^{\text{env}} Y_l^m(\cos \theta, \sin(-m\phi)) + \sum_l \sum_{m=1}^l L_{lm}^{\text{env}} Y_l^m(\cos \theta, \cos(m\phi)).$$

where Y_l^m are the spherical harmonics basis functions.

Global spherical angles ϕ and θ can be expressed as

$$\cos \theta = z, \quad \cos \phi = \frac{x}{\sqrt{1-z^2}}, \quad \sin \phi = \frac{y}{\sqrt{1-z^2}}.$$

where x, y, z are the components of the incident direction vector

$$\vec{\omega}'(u_1, u_2) = (x(u_1, u_2), y(u_1, u_2), z(u_1, u_2)).$$

The derivatives of the environment requires the derivatives of the spherical basis functions. In practice the number of bands l is quite small, thus it is worth pre-computing these basis functions in algebraic form by hand.

The first 9 spherical basis functions are obtained as:

$$\begin{aligned} Y_0^0 &= 0.282095 \\ Y_1^{-1} &= 0.488603y \\ Y_1^0 &= 0.488603z \\ Y_1^1 &= 0.488603x \\ Y_2^{-2} &= 1.092548xy \\ Y_2^{-1} &= 1.092548yz \\ Y_2^0 &= 0.315392(3z^2 - 1) \\ Y_2^1 &= 1.092548xz \\ Y_2^2 &= 0.546274(x^2 - y^2) \end{aligned}$$

The derivatives with respect to u_d ($d = 1, 2$) of the basis functions can be obtained by a direct derivation of the formulae:

$$\begin{aligned} \frac{\partial Y_0^0}{\partial u_d} &= 0 \\ \frac{\partial Y_1^{-1}}{\partial u_d} &= 0.488603 \frac{\partial y}{\partial u_d} \\ \frac{\partial Y_1^0}{\partial u_d} &= 0.488603 \frac{\partial z}{\partial u_d} \\ \frac{\partial Y_1^1}{\partial u_d} &= 0.488603 \frac{\partial x}{\partial u_d} \end{aligned}$$

$$\begin{aligned}\frac{\partial Y_2^{-2}}{\partial u_d} &= 1.092548 \left(\frac{\partial x}{\partial u_d} y + x \frac{\partial y}{\partial u_d} \right) \\ \frac{\partial Y_2^{-1}}{\partial u_d} &= 1.092548 \left(\frac{\partial y}{\partial u_d} z + y \frac{\partial z}{\partial u_d} \right) \\ \frac{\partial Y_2^0}{\partial u_d} &= 0.315392 \left(6z \frac{\partial z}{\partial u_d} \right) \\ \frac{\partial Y_2^1}{\partial u_d} &= 1.092548 \left(\frac{\partial x}{\partial u_d} z + x \frac{\partial z}{\partial u_d} \right) \\ \frac{\partial Y_2^2}{\partial u_d} &= 0.546274 \left(2x \frac{\partial x}{\partial u_d} - 2y \frac{\partial y}{\partial u_d} \right)\end{aligned}$$

References

1. S. Agarwal, R. Ramamoorthi, S. Belongie, and H. W. Jensen. Structured importance sampling of environment maps. *ACM Trans. Graph.*, 22(3):605–612, 2003.
2. J. Arvo, K. Torrance, and B. Smits. A framework for the analysis of error in global illumination algorithms. *SIGGRAPH '94 Proceedings*, pages 75–84, 1994.
3. P. Bekaert. Error control for radiosity. In *Rendering Techniques '97*, 1997.
4. M. R. Bolin and G. W. Meyer. An error metric for Monte Carlo ray tracing. In *Rendering Techniques '97*, pages 57–68, 1997.
5. D. Burke, A. Ghosh, and W. Heidrich. Bidirectional importance sampling for direct illumination. In *Rendering Techniques '05*, pages 147–156, 2005.
6. P. Clarberg, W. Jarosz, T. Akenine-Möller, and H. W. Jensen. Wavelet importance sampling: Efficiently evaluating products of complex functions. In *Proceedings of ACM SIGGRAPH 2005*. ACM Press, 2005.
7. D. Cline, P. K. Egbert, J. F. Talbot, and D. L. Cardon. Two stage importance sampling for direct lighting. In *Eurographics Symposium on Rendering (2006)*, 2006.
8. P. Debevec. Rendering synthetic objects into real scenes: Bridging traditional and image-based graphics with global illumination and high dynamic range photography. In *SIGGRAPH '98*, pages 189–198, 1998.
9. K. Dmitriev, S. Brabec, K. Myszkowski, and H.-P. Seidel. Interactive global illumination using selective photon tracing. In *EGRW '02*, pages 25–36, 2002.
10. A. Guillin, J. M. Marin, and C. P. Robert. Population Monte Carlo. *Journal of Computational and Graphical Statistics*, 13:907–929, 2004.
11. T. Hachisuka, W. Jarosz, R. P. Weistroffer, K. Dale, G. Humphreys, M. Zwicker, and H. W. Jensen. Multidimensional adaptive sampling and reconstruction for ray tracing. *ACM Trans. Graph.*, 27(3):1–10, 2008.
12. H. Igehy. Tracing ray differentials. In *SIGGRAPH '99*, pages 179–186, 1999.
13. Cs. Kelemen, L. Szirmay-Kalos, Gy. Antal, and F. Csonka. A simple and robust mutation strategy for the Metropolis light transport algorithm. *Comput. Graph. Forum*, 21(3):531–540, 2002.
14. A. Keller. Hierarchical Monte Carlo image synthesis. *Mathematics and Computers in Simulation*, 55(1–3):79–92, 2001.
15. A. Keller. Myths of computer graphics. In H. Niederreiter and D. Talay, editors, *Monte Carlo and Quasi-Monte Carlo Methods 2004*, pages 217–243, Berlin, 2006. Springer-Verlag.
16. T. Kollig and A. Keller. Efficient multidimensional sampling. *Computer Graphics Forum*, 21(3):557–563, 2002.
17. Y.-C. Lai, S. Fan, S. Chenney, and C. Dyer. Photorealistic image rendering with Population Monte Carlo energy redistribution. In *Eurographics Symposium on Rendering*, pages 287–296, 2007.
18. G. P. Lepage. An adaptive multidimensional integration program. Technical Report CLNS-80/447, Cornell University, 1980.
19. D. Lischinski, B. Smits, and D.P. Greenberg. Bounds and error estimates for radiosity. In *Computer Graphics Proceedings*, pages 67–75, 1994.
20. V. Ostromoukhov, C. Donohue, and P.-M. Jodoin. Fast hierarchical importance sampling with blue noise properties. *ACM Transactions on Graphics*, 23(3):488–498, 2004.
21. J. Rigau, M. Feixas, and M. Sbert. Refinement criteria based on f-divergences. In *EGRW '03*, pages 260–269, 2003.
22. F. Rousselle, P. Clarberg, L. Leblanc, V. Ostromoukhov, and P. Poulin. Efficient product sampling using hierarchical thresholding. In *Computer Graphics International 2008*, pages 465–474, 2008.
23. M. Sbert. Error and complexity of random walk Monte-Carlo radiosity. *IEEE Transactions on Visualization and Computer Graphics*, 3(1), 1997.
24. L. Schjoth, J. R. Frisvad, K. Erleben, and J. Sporring. Photon differentials. In *GRAPHITE '07*, pages 179–186, 2007.
25. F. Suykens and Y. D. Willems. Path differentials and applications. In *In Rendering Techniques 2001*, pages 257–268, 2001.
26. L. Szirmay-Kalos, T. Főris, L. Neumann, and B. Csébfalvi. An analysis to quasi-Monte Carlo integration applied to the transillumination radiosity method. *Computer Graphics Forum (Eurographics '97)*, 16(3):271–281, 1997.
27. L. Szirmay-Kalos and L. Szécsi. Deterministic importance sampling with error diffusion. *Computer Graphics Forum*, 28(4):1056–1064, 2009.
28. J. Talbot, D. Cline, and P. K. Egbert. Importance resampling for global illumination. In *Rendering Techniques*, pages 139–146, 2005.
29. Christiane Ulbricht, Alexander Wilkie, and Werner Purgathofer. Verification of physically based rendering algorithms. *Computer Graphics Forum*, 25(2):237–255, June 2006.
30. E. Veach and L. Guibas. Metropolis light transport. *SIGGRAPH '97 Proceedings*, pages 65–76, 1997.
31. T. Whitted. An improved illumination model for shaded display. *Communications of the ACM*, 23(6):343–349, 1980.

Construction of circular splats on analytic surfaces

M. Szilvási-Nagy

Department of Geometry, Budapest University of Technology and Economics

Abstract

In this paper we present a construction method of “circular” surface pieces – called surface splats – approximating a given surface locally. The necessary data are curvature values at specified points of the surface, which are computed from the describing vector function. Surface splats have been developed for reducing data structures in discrete surface representations. This presented algorithm can be applied also to triangle meshes or point clouds using estimated curvatures and principal directions of the approximated surface.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Surface representations, Geometric algorithms

1. Introduction

Surface splats have been constructed in different applications working with triangle meshes or point clouds. They first appeared in rendering processes. Splats were defined as piecewise linear surface primitives of elliptic shape aligned to the principal curvature directions of the surface represented by a triangle mesh. Several mesh decimation algorithms have been developed by replacing a region with a proper local approximating splat. In point-based surface representations the piecewise linear geometry of surface splats provide effective resampling strategies. A detailed survey of such methods is given in ². Local approximating quadrics have been constructed and used for mesh simplification in ¹. The necessary geometrical information is the same as here, and error analysis have been made on “synthetic” meshes generated by triangulating analytic surfaces.

In our investigation the curvatures are computed from the parameter vector equations of the sample surfaces. Applying the developed numerical methods to triangle meshes for the estimation of normal and principal curvatures (³, ⁴, ⁵), the input data of the presented construction can be obtained from discrete surface representations.

2. Computation of a circular surface splat

We assume that the surface is represented by the vector function $r(u, v)$, $(u, v) \in [u_1, u_2] \times [v_1, v_2]$, which is twice continuously differentiable allowing the computation of normal

and principal curvatures. Let P be a specified point on the surface with the position vector $r(u_p, v_p)$ and $\kappa_1 < \kappa_2$ the principal curvatures (excluding the umbilical points). The corresponding curvature radii are the reciprocal values of non-zero curvatures and are denoted by ρ_1 and ρ_2 , respectively. The principal directions are perpendicular to each other, and they determine with the surface normal the two normal planes, in which the osculating circles to the normal sections with the largest radius ρ_1 and the smallest radius ρ_2 , respectively, are lying. The circular surface splat will be constructed around the point P in the local coordinate system $e_1, e_2, e_3 = m$, where e_1 is the principal direction vector of κ_1 , e_2 is the principal direction vector of κ_2 and m is the surface normal vector, i.e. the unit vector of $r_u \times r_v$ at the parameter values (u_p, v_p) (Figure 1).

The dimensions of the constructed splat are determined by a given arc length s , measured along the osculating circles of radii ρ_1 and ρ_2 . The corresponding central half angles in these circles are denoted by α and β , respectively. If one of the principal curvatures is zero, consequently the osculating circle does not exist, then the given arc length s will be measured along the tangent line in the corresponding principal direction.

In Figure 1 the osculating circles in the normal sections determined by the two principal directions are shown. In the first principal direction the normal curvature κ_1 is negative, in the second κ_2 is positive. This is the case in a surface point of hyperbolic type.

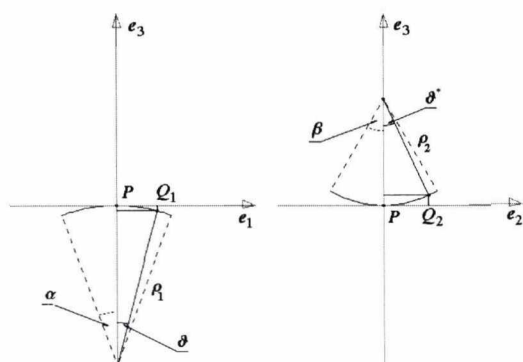


Figure 1: Osculating circles in the principal normal sections and the arbitrary points Q_1 and Q_2 on the arcs which are generating points of the surface splat.

The position vector of the point Q_1 in the local basis $\{P, e_1, e_2, m\}$ is, when $\kappa_1 < 0$

$$P\vec{Q}_1 = \rho_1 \sin \vartheta e_1 + (-\rho_1 + \rho_1 \cos \vartheta) e_3, \quad 0 \leq \vartheta \leq \alpha.$$

The position vector of Q_2 is in the case, when $\kappa_2 > 0$

$$P\vec{Q}_2 = \rho_2 \sin(\vartheta \frac{\beta}{\alpha}) e_2 + (\rho_2 - \rho_2 \cos(\vartheta \frac{\beta}{\alpha})) e_3, \quad 0 \leq \vartheta \leq \alpha.$$

As the two arc lengths are equal, $\alpha = s/\rho_1$ and $\beta = s/\rho_2$. The parameter of the points Q_1 and Q_2 is ϑ . The point Q_2 on the second arc is determined by the parameter $\vartheta\beta/\alpha$ (denoted by ϑ^* in Figure 1). If κ_1 or κ_2 is zero, then the straight line segment of the length s is computed on the corresponding tangent line, i.e. in the direction e_1 or e_2 , respectively, instead of the circular arcs.

The required surface splat is generated by rotating the point Q_1 around the surface normal of the direction vector $e_3 = m$, while correcting its third component in such a way that at the rotational angle $\varphi = \pi/2$ it becomes equal to the third component of the point Q_2 . This rotation and distortion is written in the following vector equation of the generated surface splat

$$q(\vartheta, \varphi) = p + \rho_1 \sin \vartheta \cos \varphi e_1 + \rho_2 \sin(\vartheta \frac{\beta}{\alpha}) \sin \varphi e_2 + \left((-\rho_1 + \rho_1 \cos \vartheta) \cos^2 \varphi + (\rho_2 - \rho_2 \cos(\vartheta \frac{\beta}{\alpha})) \sin^2 \varphi \right) e_3, \quad 0 \leq \vartheta \leq \alpha, \quad 0 \leq \varphi \leq 2\pi.$$

In this equation the surface point P is assumed to be of hyperbolic type. In the case of an elliptic point the third coordinate of the point Q_2 has to be computed similarly as that of the point Q_1 . In Figure 2 the boundary curve of a circular splat at a hyperbolic point is shown.

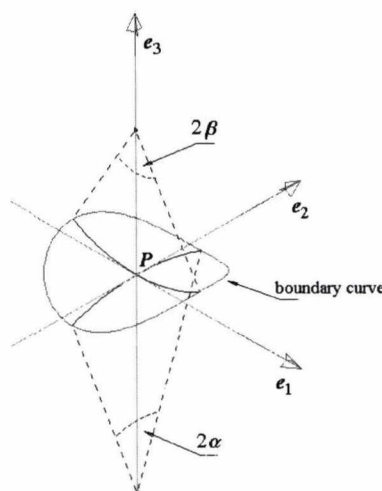


Figure 2: The boundary curve of the generated circular splat and the arcs of the two osculating circles in the principal sections.

3. Examples

In Figure 3 and in Figure 4 series of constructed circular surface splats are shown. The given arc length s is one quarter of the radius of the cylinder and the radius of the meridian circle of the torus, respectively. The points of the cylinder are of parabolic type, i.e. one of the principal curvatures is zero, the corresponding principal direction is that of the generating lines. The points on the outer torus are of elliptic and on the inner torus are of hyperbolic type, respectively.

On the boundaries of the splats series of points are shown, which have been computed for the purpose of error analysis. These are on the actual surface the projections of the end points of the osculating circle arcs in normal sections. The central angles of the circular arcs have been computed from the given arc length s (this is the "radius" of the constructed splat) and the reciprocal values of the corresponding normal curvatures, i.e. the curvature radii. The end points are determined by these central angles, then are "projected" onto the surface by finding their minimal distances to the surface. In this way a circular patch has been generated around each selected surface point. In the figures these patches are shown with 72 surface points generated in 72 normal sections.

The approximation error of the circular splat has been computed in the normal sections by the distance of the corresponding splat point and surface point. The analysis has shown that the absolute error is between 10^{-8} and $2 \cdot 10^{-3}$ when the radius of the cylinder or the torus is 10 and the splat radius is 10/3. The error is the largest in the direction of the largest principal curvature. The error reduces with smaller splat radius, and it is within the approximation error bound of the osculating circles of the normal sections to the surface.

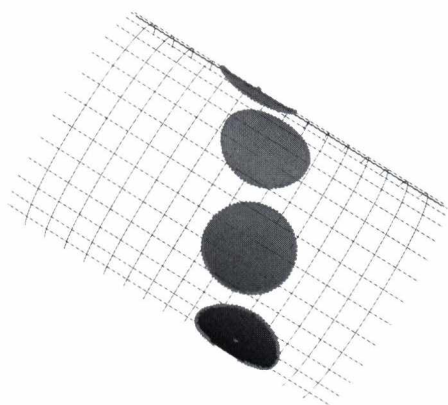


Figure 3: Surface splats on the cylinder.

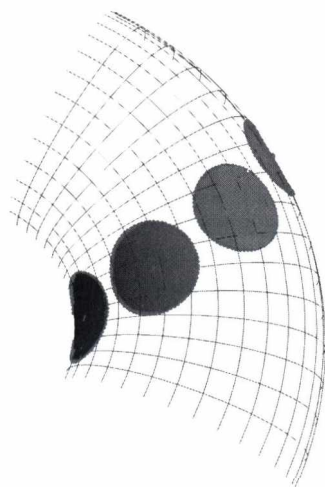


Figure 4: Surface splats on the torus.

The computations and the figures have been made with the symbolical algebraic program package Mathematica⁶. In Figure 3, 4 the colors of the surfaces and the splats are partially inverted, because the difference between the two surfaces is within the numerical tolerance value of the visibility algorithm.

4. Conclusions

We have shown an algorithm for computing a circular surface splat around a specified point of a surface represented by a parametric vector function. The input data of the computation are a given arc length (the "radius" of the splat) and the principal curvatures of the surface at a given point. The constructed surface splat can be used for replacing a circular neighborhood of the surface around the common center point. The error analysis has shown that the constructed splat approximates the surface within the same error tolerance as the osculating circles in the corresponding normal sections. Several methods have been developed for getting the curvature information from discrete surface representations, that are the input data of the presented construction. Therefore, our algorithm can be applied also to surfaces represented by triangle meshes.

References

1. P. S. Heckbert, M. Garland. Optimal triangulation and quadric-based surface simplification. *Computational geometry* 14:49-65, 1999. 1
2. L. Kobbelt, M. Botsch. A survey of point-based techniques in computer graphics. *Computers & Graphics* 28:801-814, 2004. 1
3. M. Szilvási-Nagy. About curvatures on triangle meshes. *KoG, Scientific and Professional Journal of Croatian Society for Geometry and Graphics*, 10:13-18, 2006. 1
4. M. Szilvási-Nagy. A curvature-based approach to milling path generation on triangular surfaces. IV. Magyar Számítógépes Grafika és Geometria Konferencia, Budapest, 2007. November 13-14: 64-67. 1
5. M. Szilvási-Nagy. Face-based estimations of curvatures on triangle meshes. *Journal for Geometry and Graphics*, 12:63-73, 2008. 1
6. S. Wolfram. *Mathematica. A System for Doing Mathematics by Computer*, Second Edition. Addison-Wesley, 1991. 3

Interpolation with cyclic curves and surfaces

Ágoston Róth¹ and Imre Juhász²

¹ Department of Applied Mathematics, Babeş - Bolyai University, Cluj-Napoca, Romania, e-mail: agoston.roth@math.ubbcluj.ro

² Department of Descriptive Geometry, University of Miskolc, Miskolc, Hungary, e-mail: agtji@uni-miskolc.hu

Abstract

We provide explicit formulas to interpolate a given sequence (array) of data points by cyclic curves (surfaces) with uniform parametrization. We also derive a closed formula for the area enclosed by plane cyclic curves. Moreover, we formulate a conjecture on the parametrization of interpolating cyclic plane curves of minimal area.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Nowadays, in CAGD curves are described by means of the combination of points and functions in the form

$$\begin{cases} \mathbf{g}: [a, b] \rightarrow \mathbb{R}^\delta, \delta \geq 2, \\ \mathbf{g}(u) = \sum_{j=0}^m F_j(u) \mathbf{d}_j \end{cases} \quad (1)$$

where points are \mathbf{d}_j called control points and functions $F_j(u)$ are blending functions. In applications, however we often know only points through which the curve to be designed has to pass through. This problem can be formulated as follows. Consider a sequence of points $\mathbf{p}_i \in \mathbb{R}^\delta, (i = 0, 1, \dots, m)$ called data points, and associated parameter values

$$a \leq u_0 < u_1 < \dots < u_m \leq b.$$

Hereafter, we will refer to the sequence

$$\{(u_i, \mathbf{p}_i)\}_{i=0}^m \in \mathcal{M}_{1,m+1}([a, b] \times \mathbb{R}^\delta)$$

as nodes. The task is to find control points $\mathbf{d}_j \in \mathbb{R}^\delta, (j = 0, 1, \dots, m)$ for which the interpolating conditions

$$\mathbf{g}(u_i) = \mathbf{p}_i, (i = 0, 1, \dots, m)$$

hold. This problem is equivalent with the solution of the linear system

$$\begin{bmatrix} F_0(u_0) & F_1(u_0) & \cdots & F_m(u_0) \\ F_0(u_1) & F_1(u_1) & \cdots & F_m(u_1) \\ \vdots & \vdots & \ddots & \vdots \\ F_0(u_m) & F_1(u_m) & \cdots & F_m(u_m) \end{bmatrix} \begin{bmatrix} \mathbf{d}_0 \\ \mathbf{d}_1 \\ \vdots \\ \mathbf{d}_m \end{bmatrix} = \begin{bmatrix} \mathbf{p}_0 \\ \mathbf{p}_1 \\ \vdots \\ \mathbf{p}_m \end{bmatrix} \quad (2)$$

that always has a unique solution provided that functions $\{F_j\}_{j=0}^m$ are linearly independent, i.e., the system $\{F_j\}_{j=0}^m$ is a basis of a vector space of functions.

Specification of data points \mathbf{p}_i is intuitive and an easy to use tool for a designer. Parameter values u_i , however have no direct geometric meaning, although they have a significant influence on the shape of the interpolating curve. In ¹ one can find various methods for the specification of these values.

In this paper we show some characteristics of interpolating cyclic curves, which is a curve description method of type (1) and has recently been introduced.

2. Preliminaries

The authors of ³ introduced a new method for closed curve and surface modeling. The method is based on the cyclic basis

$$\left\{ C_{i,n}(u) = \frac{c_n}{2^n} (1 + \cos(u + i\lambda_n))^n, u \in [\mu, \mu + 2\pi] \right\}_{i=0}^{2n} \quad (3)$$

of the vector space

$$\mathcal{V}_n = \langle 1, \cos(u), \sin(u), \dots, \cos(nu), \sin(nu) \rangle$$

of trigonometric polynomials of degree at most $n \geq 1$, where $\lambda_n = \frac{2\pi}{2n+1}, \mu \in \mathbb{R}$ is arbitrarily fixed, and the normalizing constant

$$c_n = \frac{2^{2n}}{(2n+1) \binom{2n}{n}} \quad (2)$$

fulfills the recursion

$$\begin{cases} c_1 = \frac{2}{3}, \\ c_n = \frac{2n}{2n+1} c_{n-1}, n \geq 2. \end{cases}$$

By means of these basis functions, we can define the cyclic curve.

Definition 1 (Cyclic curve) The curve

$$\mathbf{a}_n(u) = \sum_{i=0}^{2n} C_{i,n}(u) \mathbf{d}_i, u \in [-2\pi, 0], \quad (4)$$

is called cyclic curve of degree $n \geq 1$ that is uniquely determined by its control polygon

$$\mathbf{D}_n = [\mathbf{d}_0, \mathbf{d}_1, \dots, \mathbf{d}_{2n}] = [\mathbf{d}_i]_{i=0}^{2n} \in \mathcal{M}_{1,2n+1}(\mathbb{R}^\delta), \delta \geq 2,$$

and basis functions (3).

This type of curves possesses the following advantageous properties:

- singularity free parametrization (the curve is of C^∞ continuity at all regular points and at singular points non-vanishing left and right derivatives exist);
- convex hull property;
- cyclic symmetry (the shape of the curve does not change when its control points are cyclically permuted);
- closure for the affine transformation of their control points;
- pseudo local controllability;
- variation diminishing;
- an efficient closed formula for the degree elevation from n to $n+r, r \geq 1$ which results in a sequence of control polygons that converge to the cyclic curve.

Definition 2 (Cyclic surface) The surface

$$\mathbf{s}_{n,m}(u, v) = \sum_{i=0}^{2n} \sum_{j=0}^{2m} \mathbf{d}_{ij} C_{i,n}(u) C_{j,m}(v), \quad (5)$$

$$(u, v) \in [-2\pi, 0] \times [-2\pi, 0], \quad (6)$$

is called cyclic surface of degree (n, m) ($n \geq 1, m \geq 1$) that is uniquely determined by its control net

$$\mathbf{D}_{n,m} = [\mathbf{d}_{ij}]_{i=0, j=0}^{2n, 2m} \in \mathcal{M}_{2n+1, 2m+1}(\mathbb{R}^3)$$

and basis functions $C_{i,n}(u)$ and $C_{j,m}(v)$ defined by (3).

With the exception of variation diminishing property, all advantageous properties of the cyclic curves are inherited by the cyclic surfaces.

Paper⁴ has specified control point configurations that result an exact description of those closed curves and surfaces the coordinate functions of which are (separable) trigonometric polynomials of finite degree. The main theoretical results of the cited article, that we will use in the subsequent sections, are the lemmas and theorems formulated below. Moreover, in⁴ it has also been shown that higher order (mixed partial) derivatives of cyclic curves/surfaces are also

cyclic curves/surfaces, and we have described the connection between the cyclic and Fourier bases of the vector space of trigonometric polynomials of finite degree.

A wide range of closed curves (like ellipses (circles), epicycloids, hypocycloids, Lissajous curves, torus knots, foliums, etc.) can be described in the form

$$\begin{cases} \mathbf{x} : [-2\pi, 0] \rightarrow \mathbb{R}^\delta, \delta \geq 2, \\ \mathbf{x}(u) = [x_1(u) \ x_2(u) \ \dots \ x_\delta(u)]^T, \end{cases} \quad (7)$$

where

$$\begin{aligned} x_l(u) &\doteq x_l\left(u, \left\{ \left(\alpha_p^l, \psi_p^l \right) \right\}_{p \in P_l}, \left\{ \left(\beta_q^l, \phi_q^l \right) \right\}_{q \in Q_l} \right) \\ &= \sum_{p \in P_l} \alpha_p^l \cos(pu + \psi_p^l) + \sum_{q \in Q_l} \beta_q^l \sin(qu + \phi_q^l), \end{aligned} \quad (8)$$

($l = 1, 2, \dots, \delta$) and $P_l, Q_l \subset \mathbb{N}, \alpha_p^l, \beta_q^l, \psi_p^l, \phi_q^l \in \mathbb{R}$. Article⁴ gave control point configurations that exactly describe such kind of closed parametric curves. All results on the class of curves (7) are based on the next lemma.

Lemma 1 (Core property of class (7)) Consider the closed curve (7) and let

$$n \geq n_{\min} = \max \left\{ e \mid e \in \cup_{l=1}^{\delta} (P_l \cup Q_l) \right\}.$$

Introduce the control points

$$\mathbf{d}_i = [x_1(-i\lambda_n) \ x_2(-i\lambda_n) \ \dots \ x_\delta(-i\lambda_n)]^T, \quad (9)$$

$$(i = 0, 1, \dots, 2n),$$

and the cyclic curve

$$\begin{aligned} \mathbf{a}_n(u) &= \sum_{i=0}^{2n} C_{i,n}(u) \mathbf{d}_i \\ &= [a_1(u) \ a_2(u) \ \dots \ a_\delta(u)]^T, u \in [-2\pi, 0] \end{aligned}$$

defined by these control points. In this case the l th ($l = 1, 2, \dots, \delta$) coordinate function of \mathbf{a}_n can be expressed as

$$\begin{aligned} a_l(u) &= \frac{1}{\binom{2n}{n}} \left(\sum_{p \in P_l} \alpha_p^l \binom{2n}{n-p} \cos(pu + \psi_p^l) \right. \\ &\quad \left. + \sum_{q \in Q_l} \beta_q^l \binom{2n}{n-q} \sin(qu + \phi_q^l) \right). \end{aligned} \quad (10)$$

Vice versa, if the coordinate functions of a cyclic curve are of the form (10), then its control points are exactly the points (9).

The direct application of Lemma 1 provides a ready to use tool for the control point based exact description of curves (7) by means of cyclic curves. The method is formulated in the next theorem.

Theorem 1 (Exact description of curve class (7)) Consider the closed curve

$$\begin{aligned} \tilde{\mathbf{x}}(u) &= [\tilde{x}_1(u) \ \tilde{x}_2(u) \ \dots \ \tilde{x}_\delta(u)]^T, \\ u &\in [-2\pi, 0], \end{aligned}$$

the coordinate functions of which are

$$\tilde{x}_l(u) = \sum_{p \in P_l} \tilde{\alpha}_p^l \cos(pu + \tilde{\psi}_p^l) + \sum_{q \in Q_l} \tilde{\beta}_q^l \sin(qu + \tilde{\varphi}_q^l), \quad (l = 1, 2, \dots, \delta), \quad (11)$$

where $P_l, Q_l \subset \mathbb{N}$ and $\tilde{\alpha}_p^l, \tilde{\beta}_q^l, \tilde{\psi}_p^l, \tilde{\varphi}_q^l \in \mathbb{R}$. Let

$$n \geq n_{\min} = \max \left\{ e \mid e \in \cup_{l=1}^{\delta} (P_l \cup Q_l) \right\}$$

and $r \in \mathbb{N}$. If we set constants that are in the representation of curve \mathbf{x} in Lemma 1 as

$$\alpha_p^l(n) = \frac{\binom{2n}{n}}{\binom{2n}{n-p}} p^r \tilde{\alpha}_p^l, \quad \psi_p^l(n) \equiv \tilde{\psi}_p^l + \frac{r\pi}{2}, \quad p \in P_l, \quad (12)$$

$$\beta_q^l(n) = \frac{\binom{2n}{n}}{\binom{2n}{n-q}} q^r \tilde{\beta}_q^l, \quad \varphi_q^l(n) \equiv \tilde{\varphi}_q^l + \frac{r\pi}{2}, \quad q \in Q_l, \quad (13)$$

then for all $n \geq n_{\min}$ the cyclic curve generated by control points (9) describes exactly the r th order derivative of the curve $\tilde{\mathbf{x}}$, i.e., the l th coordinate function of cyclic curve \mathbf{a}_n fulfills the equality

$$a_l(u) = \frac{d^r}{du^r} \tilde{x}^l(u) = \sum_{p \in P_l} p^r \alpha_p^l \cos(pu + \psi_p^l + \frac{r\pi}{2}) + \sum_{q \in Q_l} q^r \beta_q^l \sin(qu + \varphi_q^l + \frac{r\pi}{2}) \quad \forall u \in [-2\pi, 0], \quad (l = 1, 2, \dots, \delta)$$

Paper ⁴ also proposed control point configurations for the exact description of a class of closed surfaces (such as surfaces of revolution obtained by rotating a curve of type (7) about an axis and the Roman surface of Steiner) by means of cyclic surfaces.

3. Interpolating cyclic curves

Certainly, interpolation with cyclic curves is also possible by means of the solution of the system (2), with settings $m = 2n$ and $F_j(u) = C_{j,n}(u)$, $u \in [\mu, \mu + 2\pi]$, ($j = 0, 1, \dots, 2n$). However, if the parameter values $\{u_j\}_{j=0}^{2n}$ are equally/uniformly spaced within the interval $[\mu, \mu + 2\pi]$ there is no need for any numerical method for the solution of the system, since we can provide an explicit formula for the unknown control points \mathbf{d}_j , ($j = 0, 1, \dots, 2n$). In order to do this, at first we show a property of the control points of cyclic curves. Its proof is based on trigonometric identities

$$\cos \alpha \cos \beta = \frac{\cos(\alpha + \beta) + \cos(\alpha - \beta)}{2}, \quad (14)$$

$$\cos \alpha \sin \beta = \frac{\sin(\beta + \alpha) + \sin(\beta - \alpha)}{2}, \quad (15)$$

and

$$\sum_{i=0}^n \cos(\varphi + i\alpha) = \frac{\sin \frac{(n+1)\alpha}{2} \cos(\varphi + \frac{n\alpha}{2})}{\sin \frac{\alpha}{2}}. \quad (16)$$

(From hereon, a number in parenthesis above the equality sign indicates that we apply the corresponding trigonometric identity.)

Theorem 2 Control points \mathbf{d}_j of cyclic curve (4) are on the curve

$$\tilde{\mathbf{x}}_n : [-2\pi, 0] \rightarrow \mathbb{R}^{\delta},$$

$$\tilde{\mathbf{x}}_n(u) = \frac{1}{2n+1} \sum_{i=0}^{2n} \mathbf{d}_i + \frac{2}{2n+1} \sum_{k=0}^{n-1} \sum_{i=0}^{2n} \cos((n-k)u + \frac{2(n-k)i\pi}{2n+1}) \mathbf{d}_i$$

at equally spaced parameter values, i.e.,

$$\mathbf{d}_j = \tilde{\mathbf{x}}_n \left(-\frac{2j\pi}{2n+1} \right), \quad (j = 0, 1, \dots, 2n).$$

Proof 1 Points of the curve $\tilde{\mathbf{x}}_n(u)$ at parameter values

$$u_j = -\frac{2j\pi}{2n+1}, \quad (j = 0, 1, \dots, 2n)$$

can be written in the form

$$\begin{aligned} & \frac{1}{2n+1} \sum_{i=0}^{2n} \mathbf{d}_i \\ & + \frac{2}{2n+1} \sum_{i=0}^{2n} \sum_{k=0}^{n-1} \cos\left((n-k)\left(-\frac{2j\pi}{2n+1} + \frac{2i\pi}{2n+1}\right)\right) \mathbf{d}_i \\ & = \frac{1}{2n+1} \mathbf{d}_j + \frac{1}{2n+1} \sum_{i=0, i \neq j}^{2n} \mathbf{d}_i + \frac{2n}{2n+1} \mathbf{d}_j \\ & + \frac{2}{2n+1} \sum_{i=0, i \neq j}^{2n} \sum_{k=0}^{n-1} \cos\left((n-k)\left(-\frac{2j\pi}{2n+1} + \frac{2i\pi}{2n+1}\right)\right) \mathbf{d}_i \\ & = \mathbf{d}_j + \frac{1}{2n+1} \sum_{i=0, i \neq j}^{2n} \mathbf{d}_i \\ & + \frac{2}{2n+1} \sum_{i=0, i \neq j}^{2n} \sum_{k=0}^{n-1} \cos\left((n-k)\left(-\frac{2j\pi}{2n+1} + \frac{2i\pi}{2n+1}\right)\right) \mathbf{d}_i \\ & = \mathbf{a}_j + \frac{1}{2n+1} \sum_{i=0, i \neq j}^{2n} \mathbf{d}_i \\ & + \frac{2}{2n+1} \sum_{i=0, i \neq j}^{2n} \sum_{k=0}^{n-1} \cos\left(\frac{2n(i-j)\pi}{2n+1} + \frac{2k(j-i)\pi}{2n+1}\right) \mathbf{d}_i \end{aligned}$$

$$\stackrel{(16)}{=} \mathbf{d}_j + \frac{1}{2n+1} \sum_{i=0, i \neq j}^{2n} \mathbf{d}_i + \frac{2}{2n+1} \sum_{i=0, i \neq j}^{2n} \frac{\sin\left(\frac{n(j-i)\pi}{2n+1}\right) \cos\left(\frac{(n+1)(i-j)\pi}{2n+1}\right)}{\sin\left(\frac{(j-i)\pi}{2n+1}\right)} \mathbf{d}_i$$

$$\stackrel{(14)}{=} \mathbf{d}_j + \frac{1}{2n+1} \sum_{i=0, i \neq j}^{2n} \mathbf{d}_i + \frac{1}{2n+1} \sum_{i=0, i \neq j}^{2n} \frac{-\sin\left(\frac{(j-i)\pi}{2n+1}\right) + \sin(\pi(j-i))}{\sin\left(\frac{(j-i)\pi}{2n+1}\right)} \mathbf{d}_i = \mathbf{d}_j + \frac{1}{2n+1} \sum_{i=0, i \neq j}^{2n} \mathbf{d}_i - \frac{1}{2n+1} \sum_{i=0, i \neq j}^{2n} \mathbf{d}_i = \mathbf{d}_j.$$

Finally, the combined use of Theorems 1 and 2 gives a method to interpolate a given set of data points by a cyclic curve with uniform parametrization as follows.

Corollary 1 (Uniform curve interpolation) For given nodes

$$\left\{ \left(u_j = -\frac{2j\pi}{2n+1}, \mathbf{p}_j \right) \right\}_{j=0}^{2n} \in \mathcal{M}_{1,2n+1} \left([-2\pi, 0] \times \mathbb{R}^\delta \right),$$

the closed trigonometric curve

$$\begin{cases} \tilde{\mathbf{x}}_n : [-2\pi, 0] \rightarrow \mathbb{R}^\delta \\ \tilde{\mathbf{x}}_n(u) = \frac{1}{2n+1} \sum_{i=0}^{2n} \mathbf{p}_i + \frac{2}{2n+1} \sum_{i=0}^{2n} \sum_{k=0}^{n-1} \cos\left((n-k)u + \frac{2(n-k)i\pi}{2n+1} \right) \mathbf{p}_i \end{cases} \quad (17)$$

fulfills the interpolating conditions

$$\tilde{\mathbf{x}}_{n,0}(u_j) = \mathbf{p}_j, \quad (j = 0, 1, \dots, 2n).$$

By means of Theorem 1 we know that the curve

$$\mathbf{x}_n(u) = \frac{1}{2n+1} \sum_{i=0}^{2n} \mathbf{p}_i + \frac{2}{2n+1} \sum_{k=0}^{n-1} \sum_{i=0}^{2n} \cos\left((n-k)u + \frac{2(n-k)i\pi}{2n+1} \right) \mathbf{p}_i$$

contains the control points of the cyclic representation of the trigonometric curve (17), i.e., the pending control points of the cyclic interpolating curve are

$$\mathbf{d}_j = \mathbf{x}_n(u_j), \quad (j = 0, 1, \dots, 2n).$$

This type of interpolation is illustrated in Figure 1.

Remark 1 (Uniform surface interpolation) Based on Theorem 14 of article 4, Corollary 1 can easily be extended to cyclic surfaces (5) that must interpolate a net of data points associated with a uniform in the parameter domain.

Figure 2 illustrates a (1, 2) degree interpolating surface of uniform parametrization.

3.1. A conjecture on area constrained interpolating cyclic curves

Definition 3 (Simple curve) A curve without self-intersections (loops) is called simple.

Let us consider the simple plane curve $\mathbf{g}(u), u \in [a, b]$. The signed area of the plane figure bounded by this curve and the straight line segments that join the endpoints of this curve with the origin is

$$A[\mathbf{g}] = \frac{1}{2} \int_a^b (\mathbf{g}(u) \wedge \dot{\mathbf{g}}(u)) du,$$

where $\mathbf{v} \wedge \mathbf{w}$ denotes the third component of the cross product of vectors \mathbf{v} and \mathbf{w} . This is a corollary of Green's theorem 2.

Remark 2 If the simple plane curve $\mathbf{g}(u), u \in [a, b]$ is orientated counterclockwise, then $A[\mathbf{g}] > 0$. In what follows, we will assume this orientation.

If the curve $\mathbf{g}(u)$ is given by (1), then

$$A[\mathbf{g}] = \frac{1}{2} \int_a^b \left(\sum_{i=0}^m F_i(u) \mathbf{d}_i \right) \wedge \left(\sum_{j=0}^m \dot{F}_j(u) \mathbf{d}_j \right) du = \frac{1}{2} \sum_{i=0}^m \sum_{j=0}^m (\mathbf{d}_i \wedge \mathbf{d}_j) \int_a^b F_i(u) \dot{F}_j(u) du.$$

Since $\mathbf{d}_i \wedge \mathbf{d}_j = -\mathbf{d}_j \wedge \mathbf{d}_i$ and $\mathbf{d}_i \wedge \mathbf{d}_i = 0$, we can reduce the number of terms in the above sum, and we obtain

$$A[\mathbf{g}] = \frac{1}{2} \sum_{i=0}^{m-1} \sum_{j=i+1}^m (\mathbf{d}_i \wedge \mathbf{d}_j) \int_a^b (F_i(u) \dot{F}_j(u) - F_j(u) \dot{F}_i(u)) du. \quad (18)$$

The next proposition provides a closed form of the formula (18) in the case of simple cyclic plane curves.

Proposition 1 (Area bounded by simple plane cyclic curves) The signed area of the simple plane cyclic curve (4) is

$$A[\mathbf{a}_n] = \frac{4\pi}{(2n)^2 (2n+1)^2} \sum_{i=0}^{2n-1} \sum_{j=i+1}^{2n} (\mathbf{d}_i \wedge \mathbf{d}_j) \cdot \left(\sum_{k=0}^{n-1} \binom{2n}{k}^2 (n-k) \sin\left(\frac{2(n-k)(i-j)\pi}{2n+1} \right) \right).$$

Proof 2 We know that the i th ($i = 0, 1, \dots, 2n$) cyclic basis function can be written in the form

$$C_{i,n}(u) = \frac{1}{2n+1} + \frac{2}{(2n+1) \binom{2n}{n}} \cdot \sum_{k=0}^{n-1} \binom{2n}{k} \cos\left((n-k)u + \frac{2(n-k)i\pi}{2n+1} \right).$$

Thus, its first order derivative is

$$\frac{d}{du} C_{i,n}(u) = \dot{C}_{i,n}(u) =$$

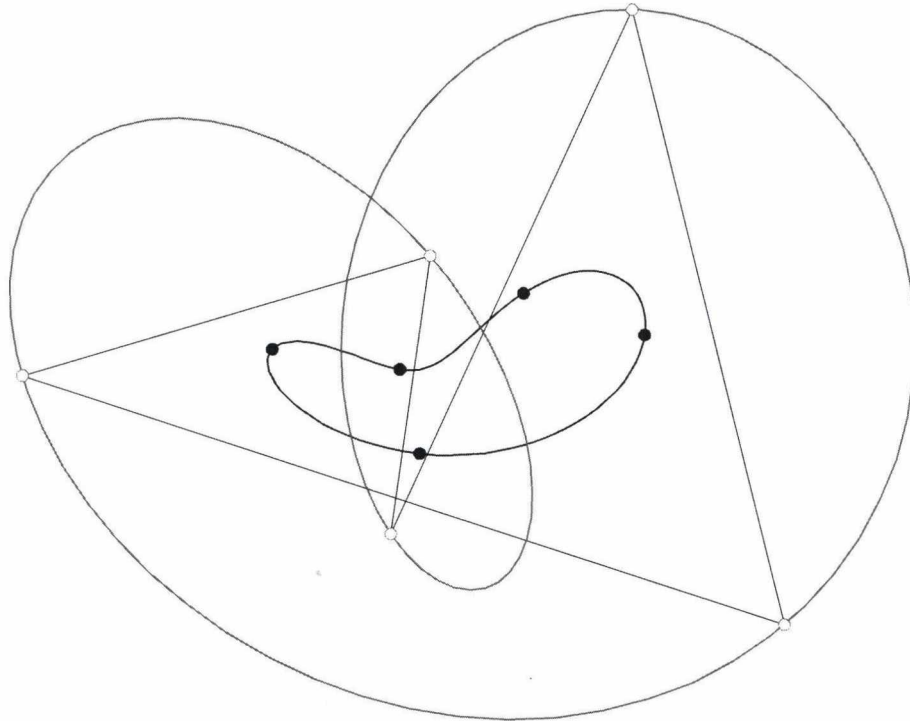


Figure 1: An interpolating cyclic curve (blue) of degree two and the locus of its control points (red curve) (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

$$-\frac{2}{(2n+1) \binom{2n}{n}} \sum_{k=0}^{n-1} \binom{2n}{k} (n-k) \sin \left((n-k)u + \frac{2(n-k)i\pi}{2n+1} \right).$$

In order to use the formula (18), with settings $m = 2n$ and $F_i(u) = C_{i,n}(u)$, $u \in [\mu, \mu + 2\pi]$, $\mu \in \mathbb{R}$, $(i = 0, 1, \dots, 2n)$, we have to evaluate the integrals

$$I_{ij} = \int_{\mu}^{\mu+2\pi} (C_{i,n}(u) \dot{C}_{j,n}(u) - C_{j,n}(u) \dot{C}_{i,n}(u)) du,$$

where $i = 0, 1, \dots, 2n-1$ and $j = i+1, i+2, \dots, 2n$. Observe that

$$I_{ij} = \frac{2}{(2n+1)^2 \binom{2n}{n}} J_{ij} + \frac{2^2}{(2n+1)^2 \binom{2n}{n}^2} K_{ij},$$

where

$$J_{ij} = \sum_{k=0}^{n-1} \binom{2n}{k} (n-k) \int_{\mu}^{\mu+2\pi} \sin \left((n-k)u + \frac{2(n-k)i\pi}{2n+1} \right) du - \sum_{l=0}^{n-1} \binom{2n}{l} (n-l) \int_{\mu}^{\mu+2\pi} \sin \left((n-l)u + \frac{2(n-l)j\pi}{2n+1} \right) du = 0$$

$$K_{ij} = \int_{\mu}^{\mu+2\pi} \left(\sum_{l=0}^{n-1} \binom{2n}{l} \cos \left((n-l)u + \frac{2(n-l)j\pi}{2n+1} \right) \right) \cdot \left(\sum_{k=0}^{n-1} \binom{2n}{k} (n-k) \sin \left((n-k)u + \frac{2(n-k)i\pi}{2n+1} \right) \right) du - \int_{\mu}^{\mu+2\pi} \left(\sum_{k=0}^{n-1} \binom{2n}{k} \cos \left((n-k)u + \frac{2(n-k)i\pi}{2n+1} \right) \right) \cdot \left(\sum_{l=0}^{n-1} \binom{2n}{l} (n-l) \sin \left((n-l)u + \frac{2(n-l)j\pi}{2n+1} \right) \right) du.$$

Now, using trigonometric identity (15) and the simple equation

$$\int_{\mu}^{\mu+2\pi} \cos(ku + \varphi) \sin(lu + \lambda) du = \begin{cases} 0, & k \neq l, \\ \pi \sin(\lambda - \varphi), & k = l, \end{cases}$$

(for arbitrary but fixed $k, l \in \mathbb{N}$ and $\varphi, \lambda \in \mathbb{R}$ numbers), we conclude that

$$K_{ij} = 2\pi \sum_{k=0}^{n-1} \binom{2n}{k}^2 (n-k) \sin \left(\frac{2(n-k)(i-j)\pi}{2n+1} \right).$$

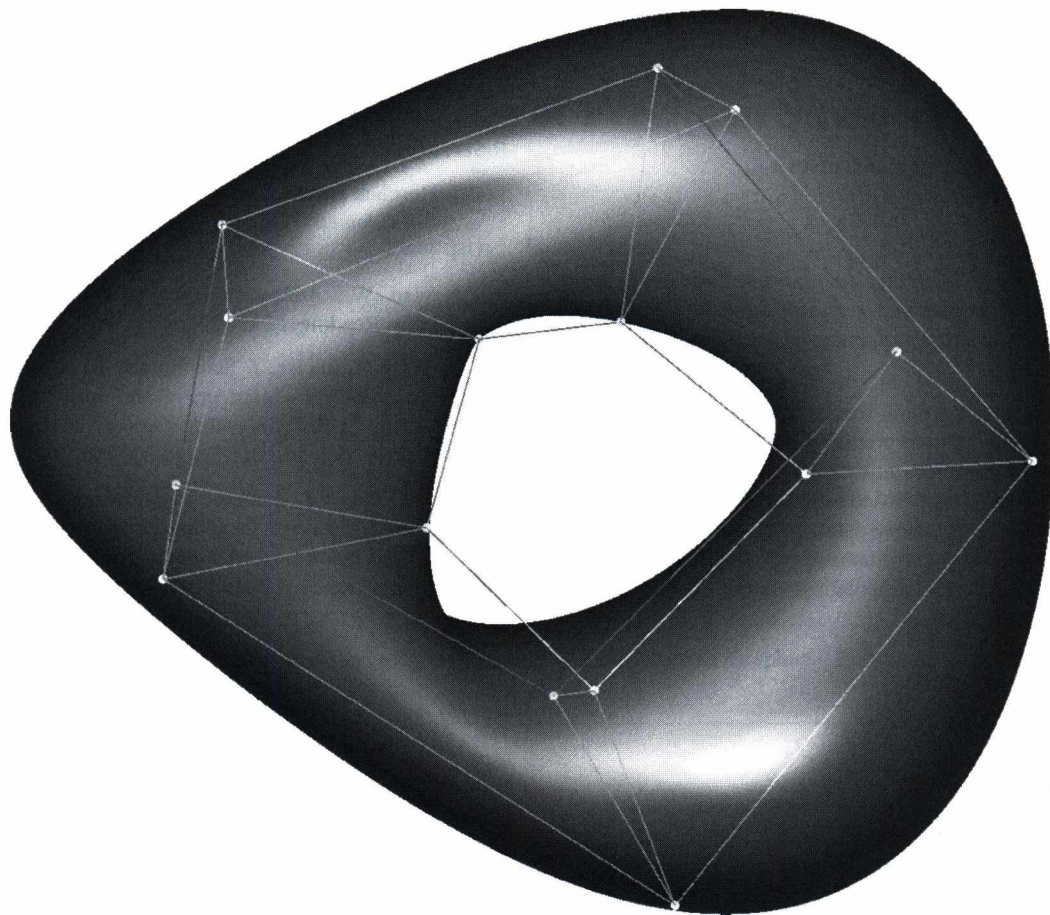


Figure 2: A (1,2) degree interpolating surface of uniform parametrization

Thus, finally we obtain that

$$I_{ij} = \frac{2^3 \pi}{(2n+1)^2 \binom{2n}{n}^2} \sum_{k=0}^{n-1} \binom{2n}{k}^2 (n-k) \sin\left(\frac{2(n-k)(i-j)\pi}{2n+1}\right)$$

from which automatically follows the pending formula of the signed area

$$\begin{aligned} A[\mathbf{a}_n] &= \frac{1}{2} \sum_{i=0}^{2n-1} \sum_{j=i+1}^{2n} (\mathbf{d}_i \wedge \mathbf{d}_j) \cdot I_{ij} \\ &= \frac{4\pi}{(2n+1)^2 \binom{2n}{n}^2} \sum_{i=0}^{2n-1} \sum_{j=i+1}^{2n} (\mathbf{d}_i \wedge \mathbf{d}_j) \cdot \left(\sum_{k=0}^{n-1} \binom{2n}{k}^2 (n-k) \sin\left(\frac{2(n-k)(i-j)\pi}{2n+1}\right) \right), \end{aligned}$$

which completes the proof.

Conjecture 1 Consider the parameter vector $\mathbf{u} = \{u_i\}_{i=0}^{2n}$ such that

$$0 = u_0 > u_1 > \dots > u_{2n} = -2\pi$$

and the nodes

$$\{(u_i, \mathbf{p}_i)\}_{i=0}^{2n} \in \mathcal{M}_{1,2n+1}([-2\pi, 0] \times \mathbb{R}^2).$$

We have already shown, that solving the linear system (2) (with settings $m = 2n$ and $F_i(u) = C_{i,n}(u)$, $u \in [-2\pi, 0]$, ($i = 0, 1, \dots, 2n$)) we obtain the control polygon

$$\mathbf{D}_{n|\mathbf{u}} = [\mathbf{d}_{i|\mathbf{u}}]_{i=0}^{2n} \in \mathcal{M}_{1,2n+1}(\mathbb{R}^2)$$

that generates the plane cyclic curve

$$\mathbf{a}_{n|\mathbf{u}}(u) = \sum_{i=0}^{2n} \mathbf{d}_{i|\mathbf{u}} C_{i,n}(u)$$

that fulfills the interpolating conditions

$$\mathbf{a}_{n|\mathbf{u}}(u_i) = \mathbf{p}_i, \quad (i = 0, 1, \dots, 2n).$$

Then, the area functional $A[\mathbf{a}_n|\mathbf{u}]$ attains its global minimum at the parameter vector

$$\mathbf{u} = \{u_i\}_{i=0}^{2n} = \{-i\lambda_n\}_{i=0}^{2n} = \left\{ -\frac{2\pi i}{2n+1} \right\}_{i=0}^{2n}$$

for self-intersection (loop) free curves $\mathbf{a}_n|\mathbf{u}$.

4. Conclusions

We have briefly reviewed the notions and basic properties of closed cyclic curves and surfaces. As new results, we have demonstrated new properties of suchlike curves and surfaces:

- we have provided explicit formulas to interpolate a given sequence (array) of data points by cyclic curves (surfaces) with uniform parametrization;
- we have also derived a closed formula for the area of plane figures enclosed by simple plane cyclic curves;
- moreover, we have formulated a conjecture on the parametrization of interpolating simple cyclic plane curves of minimal area.

5. Acknowledgements

Á. Róth was supported by the Hungarian University Federation of Cluj Napoca (2009/2010).

References

1. J. Hoschek and D. Lasser. *Fundamentals of Computer Aided Geometric Design*. AK Peters, Wellesley, 1993. 1
2. Kaplan, W. *Advanced Calculus*, 4th ed. Addison-Wesley, Reading, MA, 1991. 4
3. Á. Róth, I. Juhász, J. Schicho and M. Hoffmann. A cyclic basis for closed curve and surface modeling. *Computer Aided Geometric Design*, **26**(5), pp. 528–546, 2009. doi: 10.1016/j.cagd.2009.02.002. 1
4. Á. Róth and I. Juhász. Control point based exact description of a class of closed curves and surfaces. *Computer Aided Geometric Design*, doi:10.1016/j.cagd.2009.11.005 (accepted) 2009. 2, 3, 4

Constrained Fairing of Freeform Surfaces

Péter Salvi¹, Tamás Várady²

¹ Loránd Eötvös Science University, Budapest

² Geomagic Inc., Budapest

Abstract

New algorithms are introduced to create fair B-spline surfaces that also satisfy continuity constraints, inherited from neighboring surfaces. The proposed procedure comprises a stepwise method to assure approximate curvature continuity and adapts various existing fairing methods. A new fairing algorithm based on curvature approximation is also presented. The discussion is not restricted to four-sided patches, but an extension to handle n-sided surfaces is also provided.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

The goal of fairing is to create visually pleasing curves and surfaces in Computer Aided Geometric Design. This is also crucial in Digital Shape Reconstruction¹³, where digital models are created from millions of data points, while retaining the geometric features of scanned objects. There is no unique algebraic expression to define aesthetic requirements; however, there is a general agreement in the styling industries that even curvature distribution with large, monotone curvature areas is the most important to achieve high-quality of surfaces. For example, take reflection lines that are frequently used in the automotive industries to indicate curvature imperfections.

Approaches to surface fairing can be divided into two categories. There are *variational methods*, that simultaneously minimize least-squares distances and various smoothness functionals during surface fitting, and there are *post-processing methods* that apply changes on already existing geometries. Both methods minimize various fairness measures, and control the extent of deviations from the original data points. In many cases though, a higher priority is assigned to the fairness of the surface approximation.

1.1. Problem

There are several, traditional methods that work quite well when primary surfaces are fitted independently of their environment; however, when a complete object composed of

many connected surfaces needs to be created, not only the smoothing of the individual surfaces is needed, but the continuity between the adjacent surfaces must be taken into account. Fairing dependent connection surfaces, such as fillets or corner patches, thus poses a new problem. Several methods have been published that deal with this problem using the variational method, see for example Lai et al.⁸ or Hsu et al.⁶.

In this paper we address the problem from the post-processing perspective. Suppose that we have a model consisting of several surfaces with dependency relations as it has been defined in the functional decomposition paradigm¹³. We would like to perform fairing in a hierarchical order, first fairing the primary surfaces, then fillets, and finally corner patches. In order to achieve this, we need algorithms that create fair surfaces while assuring smooth connections to the adjacent surface elements. For fillets, smooth connection to two primary surfaces must be satisfied; for corner patches, smooth connection to n surrounding surface elements is needed. Our goal is to deal primarily with the latter problem of constrained fairing, which includes a solution for fillets, as well.

1.2. Goals

We can break this problem into two separate subproblems.

The first one is to assure continuous connections. In CAGD, continuity constraints between surfaces are typically

relaxed to *geometric continuity* instead of matching the parametric derivatives. G^1 continuity enforces common tangent planes along the shared boundaries, G^2 continuity enforces common surface curvatures. We want to tweak a surface S to match the fixed *master* surfaces M_i by *numerical* (tolerance driven) G^1 and G^2 constraints.

The second one is to perform the actual fairing. We need algorithms that smooth the surface S and retains the continuity constraints to the master surfaces. This may be as simple as doing local fairing in the “inside” region of S , or may mean special fairness measures that incorporate the constraints into the fairing process. We will evaluate different approaches later.

A combination of the above two algorithms yields a solution to our main problem.

1.3. Outline

The structure of the paper is the following. In Section 2 we briefly review some papers that are important for this work. In Section 3 a solution to the simplest configuration of four-sided corner patches is presented using an alternating sequence of constraining and fairing. The extension to n -sided patches is given in Section 4. The results are illustrated using a few examples in Section 5, and a few concluding remarks are added at the end of the paper.

2. Previous Work

There are several papers in the literature on continuity constraints and surface fairing, but a solution to constrained fairing of multiple surfaces in the post-processing context is not known to the authors. Space limitations prevent us to give a full review of all known approaches; instead we collected a few contributions that have been influential to our current project.

Concerning curvature continuity constraints — Pegna and Wolter⁹ proved the *Linkage Curve Theorem*, which gives a necessary and sufficient condition for G^2 continuity between two surfaces that share common tangent planes. The theorem states (as rephrased in Hermann et al.⁵, who also extended the theorem to G^n continuity) the following:

Two surfaces tangent along a C^1 -smooth linkage curve are curvature continuous if and only if at every point of the linkage curve, their normal curvature agrees for an arbitrary direction other than the tangent to the linkage curve.

This is the basis of our G^2 algorithm in Section 3.1.

Concerning curvature approximation — a paper by Greiner³ gives a quadratic approximation of surface curvature using a reference surface (e.g. a least-square fit of the data points). The idea here is to compute part of the Hessian matrix from the reference surface, such that the computation

still dependent on the real surface is only quadratic. Then the mean and Gauss curvatures can be computed from the Hessian. Greiner also introduces a data-dependent quadratic fairness functional, along with an algorithm minimizing this approximated curvature.

We use different fairing algorithms in our tests. These will be analyzed in more details in Section 3.2.

Knot-Removal and Reinsertion (KRR), proposed originally by Farin and Sapidis², is one of the simplest and most widely used local, control-point based fairing algorithm. It provides fair curves by removing and reinserting the *most offending knots*, thus the C^3 jumps for a cubic B-spline are reduced, which yields a nicer curve. The algorithm has a couple of variations; Hahmann⁴ extended the method to surface fairing.

Another fairing method was proposed by Salvi et al.¹¹, that approximates a smoothed *target curvature*. This is an efficient and highly flexible algorithm; however, it includes a final fitting step which may harm locality to some extent.

Finally, for n -sided corner patches (see Section 4), we cannot use bi-parametric surface algorithms. Instead, we apply a mesh based fairing algorithm described by Kobbelt et al.⁷, where the n -sided surface region is approximated by a mesh and discrete fairing is applied. A local quadratic approximation is used to compute the second-order partial derivatives and minimize the thin plate energy. Continuity constraints can be preserved at sampled data points around the boundary of the n -sided region.

3. Four-sided Patches

A B-spline surface $S(u, v)$ is defined by means of its control points P_{ij} ($i \in [0 \dots n]$, $j \in [0 \dots m]$) and its knot vectors U and V :

$$S(u, v) = \sum_{i=0}^n \sum_{j=0}^m P_{ij} N_i^{U,p}(u) N_j^{V,q}(v),$$

where p and q are the degrees of the surface in the u and v directions, respectively. The four boundaries of the surface are defined by the outermost control points — P_{0j} , P_{nj} , P_{i0} and P_{im} , respectively ($i \in [0 \dots n]$, $j \in [0 \dots m]$). In this context we call these together as the *positional frame*. Let us assume hereinafter that the outermost control points are fixed. The tangent planes at the points of the boundaries are indirectly determined by the first cross-boundary derivatives, i.e. by the inner control points P_{1j} , $P_{(n-1)j}$, P_{i1} and $P_{i(m-1)}$, respectively ($i \in [1 \dots n-1]$, $j \in [1 \dots m-1]$). We call these control points together the *tangential frame*. Finally, assume that control points of the positional and tangential frames are fixed. Then the surface curvatures at the points of the boundaries are indirectly determined by the second cross-boundary derivatives, i.e. by the inner control points P_{2j} , $P_{(n-2)j}$, P_{i2} and $P_{i(m-2)}$, respectively ($i \in [2 \dots n-2]$, $j \in [2 \dots m-2]$). We call these control points together the *curvature frame*.

It is also well-known, that when we enforce G^1 continuity independently for the individual boundaries, a so-called twist incompatibility condition must be satisfied for the mixed partial derivatives, which are indirectly determined by the twist control points $P_{11}, P_{1(m-1)}, P_{(n-1)1}$ and $P_{(n-1)(m-1)}$. After we enforce G^2 continuity for the individual boundaries, a similar compatibility condition must be satisfied to tweak the inner twist control points $P_{22}, P_{2(m-2)}, P_{(n-2)2}$ and $P_{(n-2)(m-2)}$.

Having said all these, the algorithm to set continuity constraints proceeds in the following way:

1. Insert some knots into the surface, if it has too few control points.
2. Fair the surface, while retaining C^0 continuity for each boundary, i.e. only control points within the positional frame are used for fairing.
3. Fix the positional frame and enforce G^1 continuity for each boundary.
4. Set the twist control points compatible, and repeat the G^1 computation.
5. Fair the surface, while retaining G^1 continuity, i.e. only control points within the tangential frame are used for fairing.
6. Fix the tangential frame and enforce G^2 continuity for each boundary.
7. Set the inner twist control points compatible, and repeat the G^2 computation.
8. Fair the surface, while retaining G^2 continuity, i.e. only control points within the curvature frame are used for fairing.

To sum it up, the sequence is always (i) constrain, (ii) set compatibility and (iii) apply fairing, until G^2 is achieved.

3.1. Three-Frame Method for Numerical G^1/G^2

We will treat the algorithm on a per side basis. The twists, however, need special attention, and they will be dealt with at the end of this section.

3.1.1. G^1 Continuity

There are two B-spline surfaces, M (master) and S (slave), that are (without loss of generality) joined along the $u = u_{\min}$ parameter line with C^0 continuity. We want to modify the corresponding side of the tangential frame, i.e. the second control row of S , such that the two surfaces will have numerical G^1 continuity. We also take sampled parameters v_k ($k \in [1 \dots K]$) along the border, not including the corner points. The normal vectors at these (u, v_k) points of the master surface are denoted by \underline{n}_k . If we now add displacement vectors \underline{w}_j to the control points P_{1j} ($j \in [1 \dots m-1]$), they

will modify the original tangents \underline{e}_k in the following way:

$$\hat{\underline{e}}_k = \underline{e}_k + \dot{N}_1^{U,P}(u) \sum_{j=1}^{m-1} N_j^{V,Q}(v_k) \underline{w}_j \quad (1)$$

$$= \underline{e}_k + \sum_{j=1}^{m-1} c_{k,j} \underline{w}_j. \quad (2)$$

Since we would like to avoid irrelevant control point movements, we minimize the deviation only in the surface normal direction. This leaves us with

$$\hat{\underline{e}}_k = \underline{e}_k - \underline{n}_k(\underline{e}_k \underline{n}_k). \quad (3)$$

Subtracting \underline{e}_k from (2) and (3) gives the linear equation system $A\underline{x} = \underline{b}$, where

$$A = \begin{bmatrix} c_{11} & c_{12} & \cdots & c_{1(m-1)} \\ c_{21} & c_{22} & \cdots & c_{2(m-1)} \\ \vdots & \vdots & \ddots & \vdots \\ c_{K1} & c_{K2} & \cdots & c_{K(m-1)} \end{bmatrix},$$

$$\underline{x} = \begin{bmatrix} \underline{w}_1 \\ \underline{w}_2 \\ \vdots \\ \underline{w}_{m-1} \end{bmatrix}, \quad \underline{b} = \begin{bmatrix} -\underline{n}_1(\underline{e}_1 \underline{n}_1) \\ -\underline{n}_2(\underline{e}_2 \underline{n}_2) \\ \vdots \\ -\underline{n}_K(\underline{e}_K \underline{n}_K) \end{bmatrix}.$$

We have K equations and $m-1$ unknowns. Solving the least-square equation $(A\underline{x} - \underline{b})^2 = 0$ leads to a linear system of $(A^T A)\underline{x} = A^T \underline{b}$.

3.1.2. G^2 Continuity

We have the same assumptions as in the G^1 case, but now we also have a numerical G^1 connection. Take K parameter points from the v domain: $v_k, 1 \leq k \leq K$. For every v_k , calculate the normal curvature κ_k^M of M at (u_{\min}, v_k) in the u direction. If we modify S such that its normal curvature in the u direction is the same, we will have G^2 continuity in (u_{\min}, v_k) , because of the Linkage Curve Theorem (see Section 2).

The normal curvature of a surface at (u, v) in some \underline{d} direction can be calculated as:

$$\kappa(\lambda) = \frac{L + 2M\lambda + N\lambda^2}{E + 2F\lambda + G\lambda^2},$$

where E, F, G and L, M, N are the coefficients of the first and second fundamental form, respectively; $\lambda = \frac{d_v}{d_u}$ and $\underline{d} = d_u \underline{S}_u + d_v \underline{S}_v$. In the special cases where \underline{d} is the u or v parametric direction, this is simplified into L/E and N/G , respectively.

Minimization

Instead of directly optimizing for surface curvature, we use the curvature of the surface curve $C_k(u) = S(u, v_k)$, which results in much simpler equations¹². We know from

Meusnier's theorem¹ that at a given point, the curvature κ_C of a surface curve C and the normal curvature κ of a surface S in the tangent direction of C have the following relationship:

$$\kappa = \kappa_C \cos \theta = \frac{\|\underline{C}' \times \underline{C}''\|}{\|\underline{C}'\|^3} \cos \theta = \frac{\langle \underline{C}'', \underline{n} \rangle}{\|\underline{C}'\|^2},$$

where $\underline{n} = \frac{\underline{S}_u \times \underline{S}_v}{\|\underline{S}_u \times \underline{S}_v\|}$ is the surface normal and θ is the angle between \underline{n} and the curve normal $(\underline{C}' \times \underline{C}'') \times \underline{C}'$.

Consequently, we have to solve equations of the form

$$\frac{\langle \underline{C}_k''(u_{\min}), \underline{n}_k \rangle}{\|\underline{C}_k'(u_{\min})\|^2} = \kappa_k^M. \quad (4)$$

Since

$$\begin{aligned} C_k(u) &= S(u, v_k) = \sum_{i=0}^n N_i^{U,p}(u) \left(\sum_{j=0}^m N_j^{V,q}(v_k) P_{ij} \right) \\ &= \sum_{i=0}^n N_i^{U,p}(u) \hat{P}_i, \end{aligned}$$

the first and second derivatives at the end are

$$\begin{aligned} C_k'(u_{\min}) &= \frac{p}{u_{p+1} - u_1} (\hat{P}_1 - \hat{P}_0), \\ C_k''(u_{\min}) &= \frac{p-1}{u_{p+1} - u_2} \left(\frac{p}{u_{p+2} - u_2} (\hat{P}_2 - \hat{P}_1) \right. \\ &\quad \left. - \frac{p}{u_{p+1} - u_1} (\hat{P}_1 - \hat{P}_0) \right), \end{aligned}$$

see Reference¹⁰ (assuming that $u_0 = u_1 = \dots = u_p = u_{\min}$).

If we want to modify the P_{2j} control point by a vector \underline{d}_j , we can reformulate Eq. 4 as

$$\begin{aligned} \Delta \kappa_k \|\underline{C}_k'(u_{\min})\|^2 \frac{(u_{p+1} - u_2)(u_{p+2} - u_2)}{p(p-1)} \\ = \sum_{j=2}^{m-2} N_j^{V,q}(v_k) \langle \underline{d}_j, \underline{n}_k \rangle, \end{aligned} \quad (5)$$

where $\Delta \kappa_k = \kappa_k^M - \kappa_k^S$. Note that the second half of C_k'' is eliminated in the scalar product by the (perpendicular) surface normal.

We propose two different solutions for this equation system. In the first one, in order to avoid irrelevant control point movements, the P_{2j} points are only allowed to move in an outward direction (\underline{o}_j for P_{2j}). Alternatively, we can require that the sum of the squared deviations should be minimal.

Solution by Fixed Directions

An easy and intuitive choice for outward directions is to get the cross product of the difference of the neighboring control points (Fig. 1), i.e. $\underline{o}_j = (P_{3j} - P_{1j}) \times (P_{2(j+1)} - P_{2(j-1)})$. If we define the deviation vectors \underline{d}_j as $\chi_j \underline{o}_j$ and introduce

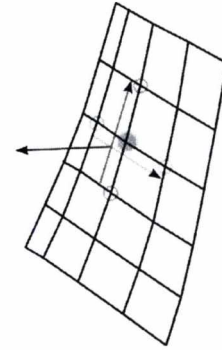


Figure 1: Outward direction

the constants α_{kj} and β_k , Eq. 5 can be rewritten as $\beta_k = \sum_{j=2}^{m-2} \alpha_{kj} \chi_j$.

Now we can create the overdetermined equation system $A\underline{x} = \underline{b}$:

$$A = \begin{bmatrix} \alpha_{12} & \alpha_{13} & \dots & \alpha_{1(m-2)} \\ \alpha_{22} & \alpha_{23} & \dots & \alpha_{2(m-2)} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{K2} & \alpha_{K3} & \dots & \alpha_{K(m-2)} \end{bmatrix},$$

$$\underline{x} = \begin{bmatrix} \chi_2 \\ \chi_3 \\ \vdots \\ \chi_{m-2} \end{bmatrix}, \quad \underline{b} = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_K \end{bmatrix}.$$

Solving the $(A^T A)\underline{x} = A^T \underline{b}$ equation results in a least-squares approximation, as earlier.

Solution by Minimal Deviation

We can also solve the equations while minimizing the squared deviation of the \hat{P}_2 control point of C_k , by requiring that it should change only in the \underline{n}_k direction. This means that Eq. 5 becomes $\beta_k \underline{n}_k = \sum_{j=2}^{m-2} N_j^{V,q}(v_k) \underline{d}_j = \sum_{j=2}^{m-2} \gamma_{kj} \underline{d}_j$.

The equation system is now $A\underline{x} = \underline{b}$, where

$$A = \begin{bmatrix} \gamma_{12} & \gamma_{13} & \dots & \gamma_{1(m-2)} \\ \gamma_{22} & \gamma_{23} & \dots & \gamma_{2(m-2)} \\ \vdots & \vdots & \ddots & \vdots \\ \gamma_{K2} & \gamma_{K3} & \dots & \gamma_{K(m-2)} \end{bmatrix},$$

$$\underline{x} = \begin{bmatrix} \underline{d}_2 \\ \underline{d}_3 \\ \vdots \\ \underline{d}_{m-2} \end{bmatrix}, \quad \underline{b} = \begin{bmatrix} \beta_1 \underline{n}_1 \\ \beta_2 \underline{n}_2 \\ \vdots \\ \beta_K \underline{n}_K \end{bmatrix}.$$

As before, $(A^T A)\underline{x} = A^T \underline{b}$ gives a least-squares approximation.

3.1.3. Twists

Every twist (or inner twist) control point of a frame has two values coming from the independent side constraints. In order to get a valid B-spline, we need a single twist control point, computed as halving of the two independent twist control points. After this, we repeat the continuity enhancement algorithm, now constraining only the $j \in [2 \dots m-2]$ control points for G^1 and the $j \in [3 \dots m-3]$ control points for G^2 , in order to obtain the best positions in accordance with the new twist values. The use of halving points is a dubious choice and optimizing the twist control points is subject of further research.

3.2. Fairing Algorithms

In this section we will look at three different algorithms suitable for our purpose. The minimal condition is that fairing should not destroy the continuity already achieved.

3.2.1. Knot Removal and Reinsertion (KRR)

This fairing method operates on the control points and we choose the one where the C^3 jump is the greatest. We perform this operation in an iterative manner several times. This retains continuity in a trivial way. On the backside, KRR is primarily suited for curve fairing. We can choose between fairing only in the u or v parametric direction — or average the two values. This simple method gives quite nice results.

3.2.2. Fairing Based on Target Curvature

We have much more flexibility in this algorithm, as the use of a target curvature enables us to define curvature continuity constraints, which is particularly valuable in our present task. However, there are some problems. The last fitting phase may harm the precision of the continuity. In this case, we need to insert another continuity enhancement step. Another drawback is that the algorithm relies on fairing the isocurves of the surface. This may cause artefacts in complex saddle surfaces.

3.2.3. Fairing Based on Curvature Approximation

We will use the curvature approximation outlined in Section 2 to enhance the previous algorithms. Given a reference surface R and a twice differentiable scalar function \tilde{h} defined on it, Greiner³ shows that the Hessian matrix of $h = \tilde{h} \circ R$ is

$$\text{Hess}_R(h) = \left(\sum_l g^{kl} (\partial_j \partial_l h - \sum_i \partial_i h \Gamma_{jl}^i) \right)_{kj},$$

where ∂_i is the partial derivative by the i^{th} argument, $\Gamma_{jl}^i = \sum_m g^{im} \langle \partial_j \partial_l R, \partial_m R \rangle$ and (g^{ij}) is the inverse of the first fundamental form of R (every index can take the values 1 and

2). Note that both Γ and g can be computed beforehand. The paper also shows that if we use the coordinate functions R_c ($c \in [1 \dots 3]$) as h , we have

$$\text{Hess}_R(R_c) = \frac{1}{EG-F^2} \begin{bmatrix} G & -F \\ -F & E \end{bmatrix} \begin{bmatrix} L & M \\ M & N \end{bmatrix} \underline{n}_c,$$

where \underline{n} is the surface normal. This has several nice properties, for example it is easy to see that

$$\sum_c \text{trace}(\text{Hess}_R(R_c))^2 = (\kappa_1^R + \kappa_2^R)^2,$$

$$\sum_c \det(\text{Hess}_R(R_c)) = \kappa_1^R \cdot \kappa_2^R.$$

The claim is that using S_c instead of R_c will result in good approximations of the curvatures of S :

$$\sum_c \text{trace}(\text{Hess}_R(S_c))^2 \approx (\kappa_1^S + \kappa_2^S)^2,$$

$$\sum_c \det(\text{Hess}_R(S_c)) \approx \kappa_1^S \cdot \kappa_2^S.$$

The reader can consult the original paper³ for more details.

In our case, we already have a very good reference surface — the original surface itself. We would like to do something similar to the algorithm in the previous section, i.e. set up a target curvature and find a surface with similar curvature. Our general scheme would be as follows:

1. Sample the original surface at intervals.
2. Compute Γ and g in these positions.
3. Set up a target curvature.
4. Minimize the deviation from the target curvature.

We can write up an equation system that depends linearly on the control points of S , so it will be easy to minimize. One option is to create a target mean curvature by smoothing the traces of Hessian matrices in the sampled points. Another alternative is to average every element of the Hessian matrices over the sampled points. Since the computation of the Hessian matrix from the control points is linear, these lead to overdefined linear equation systems, that can be solved in least-squares sense. An example is shown in Fig. 2.

As for the continuity restrictions, we can fix the outer frames as in the KRR algorithm. The advantage of this method is that it is independent of the parametric directions. However, we have to minimize a fairly large equation system, which makes its computational cost quite high.

4. n -sided Corner Patches

There are corner patches with three or more than four (usually five) connecting surfaces. The simplest representation of these is based on the so-called central split, where n quadrilateral surfaces are stitched together, see Fig. 5 for a five-sided example.

The main difficulty here is that the corner patch consists of more than one surface, so we have to ensure continuity not



(a) Before fairing



(b) After fairing

Figure 2: Isophotes of a car body panel faired by the curvature approximation method

only along the boundaries, but along the subdividing curves between the quadrilaterals as well. Moreover, for n -sided regions the previous fairing methods that were applied for bi-parametric surfaces cannot be used.

4.1. Extension of the Algorithm

In order to cope with these difficulties, we modify the procedure presented in Section 3:

1. Global fairing of the quadrilaterals (retaining C^0 or G^1 continuity on the perimeter).
2. Enforce G^1 continuity on the perimeter and between the quadrilaterals.
3. Local fairing of each quadrilateral, retaining G^1 continuity.
4. Enforce G^2 continuity on the perimeter and between the quadrilaterals.
5. Local fairing of each quadrilateral, retaining G^2 continuity.

The first step aims at creating a good base for the further operations. It can be omitted if the original surface has adequate quality. Unlike in the four-sided case, the first step here needs special attention, since multiple surfaces need to be processed simultaneously. A mesh-based fairing seems to be a natural approach.

4.2. Mesh-based Fairing

In order to avoid the parameterization problems, we discretize the corner patch, and create a triangle mesh as shown in Fig. 6. The actual fairing is done by the method suggested by Kobbelt⁷, inheriting G^1 continuity at the boundaries. Finally, we refit the surfaces using the points of the faired mesh. Although this algorithm, due to precision loss, may be inferior to the others, it creates generally a good shape and is easy to use regardless of the number of quadrilaterals involved.

5. Examples

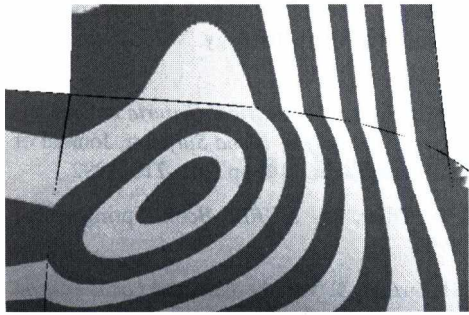
Figures 3 and 4 both show the effect of constrained fairing for four-sided corner patches. In these cases, the central surface is relatively simple; the original (a) has reasonable isophotes, but the isophote strips break when they reach the boundaries. After constrained fairing (b) the images show that numerical G^2 continuity has been achieved and fairing also nicely affected the interior without changing the master surfaces.

We have a different case in Fig. 7. Here the original corner patch (a) already had numerical G^2 that needed only very minor changes. On the other hand, fairing (b) increased the overall surface quality very much.

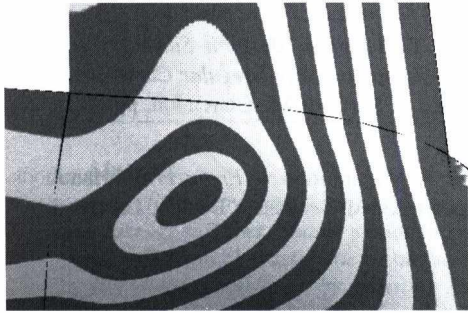
Conclusion and Future Work

A new approach that combines numerical G^2 connections between free-form surfaces with methods to create fair shapes was presented. Its primary application is to perfect connected surface elements produced in Digital Shape Reconstruction. The proposed algorithms have worked well for several models.

There is still room for improvements. For example, the algorithm that assures continuity should make a better use of the twist control points. We believe that better twist locations may significantly influence the quality of the modified surfaces. The averaging strategy of the KRR method applied for the u and v parametric directions should also be enhanced. The n -sided corner-patches should also be improved by applying such discrete fairing methods that guarantee discrete G^2 connections to the master surfaces. In fact, the discrete fairing step is considered only as a temporary solution, and alternative fairing techniques for n -sided corner patches are subject of our current investigations.

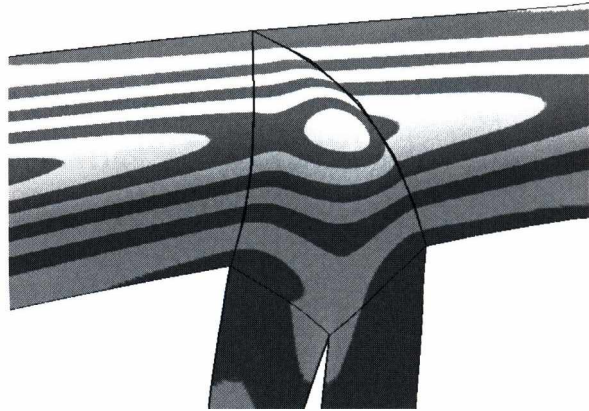


(a) Before fairing

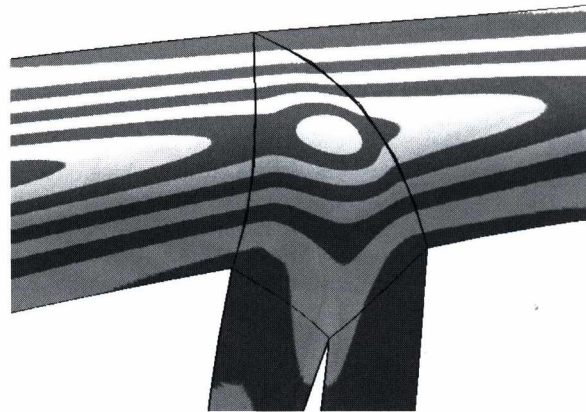


(b) After fairing

Figure 3: Fairing an X-node



(a) Before fairing



(b) After fairing

Figure 4: Fairing another X-node

References

1. G. Farin, *Curves and Surfaces for Computer Aided Geometric Design. A Practical Guide*. Academic Press, 5th Edition, 2002.
2. G. Farin, G. Rein, N. Sapidis, A. J. Worsey, *Fairing Cubic B-Spline Curves*, Computer Aided Geometric Design, Vol. 4, pp. 91–103, 1987.
3. G. Greiner, *Curvature Approximation with Application to Surface Modeling*, Teubner, pp. 241–252, 1996.
4. S. Hahmann, S. Konz, *Knot-Removal Surface Fairing Using Search Strategies*, Computer Aided Design, Vol. 30, pp. 131–138, 1998.
5. T. Hermann, G. Lukács, F-E. Wolter, *Geometrical Criteria on the higher order smoothness of composite surfaces*, Computer Aided Geometric Design, Vol. 16, pp. 907–911, 1999.
6. K.-L. Hsu, D.-M. Tsay, *Corner Blending of Free-Form N-Sided Holes*, IEEE Computer Graphics and Applications, Vol. 18, pp. 72–78, 1998.
7. L. P. Kobbelt, *Discrete Fairing and Variational Subdivision for Freeform Surface Design*, The Visual Computer, Vol. 16, pp. 142–158, 2000.
8. J.-Y. Lai, W.-D. Ueng, G^2 Continuity for Multiple

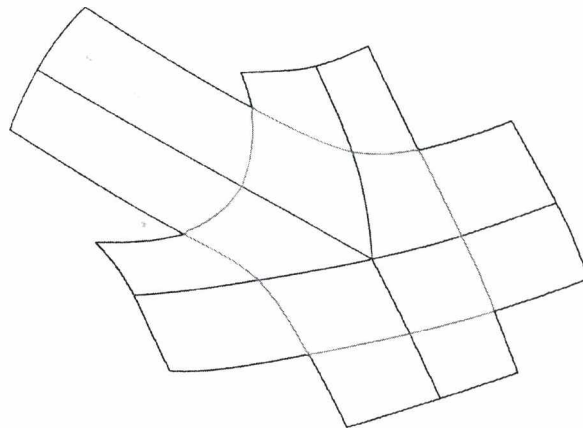


Figure 5: Five-sided corner patch consisting of five quadrilateral surfaces

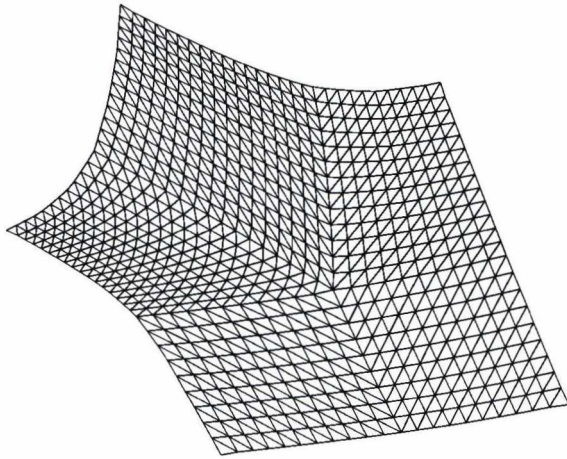
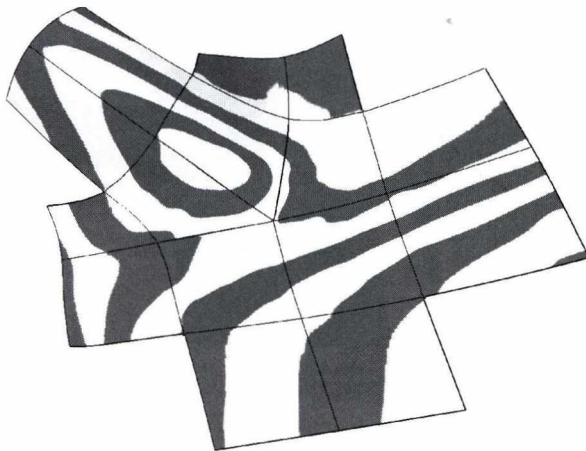
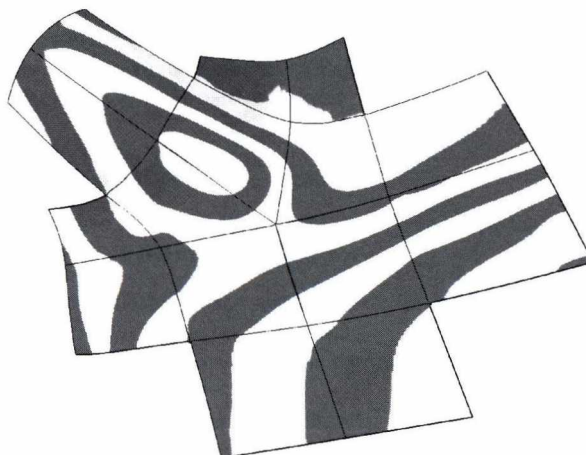


Figure 6: Triangle mesh of a five-sided corner patch



(a) Before fairing



(b) After fairing

Figure 7: Fairing a five-sided corner patch.

Surfaces Fitting, The International Journal of Advanced Manufacturing Technology, Vol. 17, pp. 575–585, 2001.

9. J. Pegna, F-E. Wolter, *Geometrical Criteria to Guarantee Curvature Continuity of Blend Surfaces*, Journal of Mechanical Design, Vol. 114, pp. 201–210, 1992.
10. L. Piegl, W. Tiller, *The NURBS Book*, Springer, 2nd Edition, 1997.
11. P. Salvi, H. Suzuki, T. Várady, *Fast and Local Fairing of B-Spline Curves and Surfaces*, Advances in Geometric Modeling and Processing, pp. 155–163, Springer, 2008.
12. T. Várady, T. Hermann, *Best Fit Surface Curvature at Vertices of Topologically Irregular Curve Networks*, Mathematics of Surfaces VI, Clarendon, pp. 411–428, 1996.
13. T. Várady, R. Martin, *Reverse Engineering*, Handbook of Computer Aided Geometric Design, Ch. 26, Elsevier, 2002.

Evaluating and Correcting the Reflection Characteristics of Surfaces

Gy. Gyurecz¹ and G. Renner²

¹Institute of Machine Design and Safety Engineering, University of Óbuda, Budapest, Hungary

²Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest, Hungary

Abstract

Reflection characteristics of surfaces, as visualized by highlight lines, are important in design of high quality (class A) surfaces. Shape, smoothness, coherence and distribution of highlight lines are good indicators of the reflection characteristic of surfaces. We present a method that enables the designer to correct the reflection status of surfaces, by adjusting its highlight lines. We developed a genetic algorithm for automatically adjusting the control points of surfaces resulting in a modified surface shape that produces the desired highlight lines. The paper discusses genetic representation, fitness function, genetic operators selection methods developed for the specific problem. The advantage of the method is its robustness and applicability to surfaces of any shapes, and any kinds of CAD representations. Effectiveness of the method for surface correction is demonstrated by several practical examples.

Keywords:

Surface correcting, Highlight lines, genetic algorithm, high quality surfaces

1. Introduction

A number of human-made objects have highly reflective surfaces: cars, airplane and ship hulls, household appliances, lamps etc. Beside geometric and manufacturability conditions, these surfaces often meet aerodynamic, hydrodynamic, and aesthetic requirements as well. The common objective is to produce smooth surfaces, without unwanted bumps and oscillations. Smoothness of a surface can be evaluated at different levels. Continuity of derivatives or curvature integrals are common tools to evaluate smoothness. Various methods (e.g. Gauss smoothing) have been developed and implemented in CAD systems for their evaluation including visual display methods.

Fairness of a surface is much more than smoothness according to the above measures. A fair surface, beyond being smooth, is free of irregularities, bumps and oscillations. Fairness can be assessed by highlight lines, the reflection curves of parallel light sources on the surface. Shape, smoothness, coherence and distribution are good indicators of surface fairness. Displaying and analysis of the complete reflection status of the surface relied on the highlight lines provides a firm basis for assessing surface quality in accordance with the aesthetic demands of the designer.

Beier and Chen in 1994¹, developed a method for computing and displaying highlight lines, but they do not address surface correction. A method for improving quality of free form surfaces by the adjustment of highlight lines was first developed by Klass and Kaufmann⁶. Correlation between highlight lines and control points is described by a non-linear equation system, which is too time consuming to solve, and the results are not always good enough.

The method developed by Zhang and Cheng introduces a great number of simplification to obtain a linear system of equation for control points to be modified through highlight lines⁷. However, the highlight line cannot accurately follow the points specified by the designer and the method yields adequate results only in a small range of the surface. Another limitation is that the method might not work when the correction needs to be carried out on the boundary of the surface.

The main problem of the above methods is represented by the complex algebraic relation between the adjusted highlight line and the control points of the corresponding surface. Our research aims to develop a method to bridge this problem. This is intended to be achieved by using a stochastic optimization method, namely genetic algorithms, which – no matter how complex the surfaces highlights are – can find modified control points even in the absence of direct algebraic relations.

The process of surface correction starts with the computation of highlight lines. Inspection of the surface quality

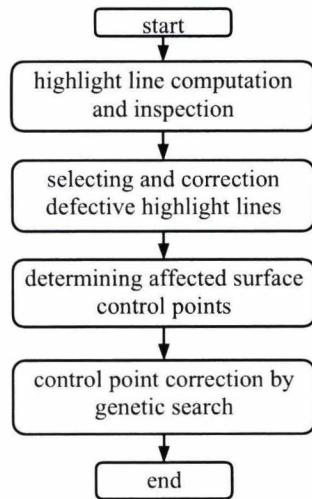


Figure 1: Block diagram of the surface correction

is carried out by using several light source settings and surface orientations. Next, the designer selects and corrects the defective highlight lines using interactive facilities of design systems. This is followed by the determination of the affected surface region and corresponding control points of the surface. Modification of the control points are performed by a genetic algorithm. Figure 1 shows the course of surface quality correction

2. Surface representation and highlight lines

Parametric representation of the free form surfaces in Bézier, B-spline or NURBS form, are widely used in CAD applications :

$$S(u, v) = \sum_{i=0}^m \sum_{j=0}^n P_{ij} N_{ik}(u) N_{jl}(v),$$

where the control points P_{ij} and the Bézier, B-spline or NURBS basis functions N_{ik} and, N_{jl} of order k and l fully determine the surface $S(u, v)$. The shape of the surface is mainly defined by control points, knots and weights of the basis functions provide additional degree of freedom in design.

Highlight lines of surfaces

A highlight line is created on the surface by the reflection of a linear lightsource of infinite length. The light source is positioned above the surface. The highlight line is a set of surface points for which the corresponding surface normal and the light source intersect each other, that is the perpendicular distance between them is zero.

Let $L(\lambda)$ denote the line of the light source:

$$L(\lambda) = A + B\lambda,$$

where A is a point on $L(\lambda)$, and B is a vector defining the direction of the line (Figure 2).

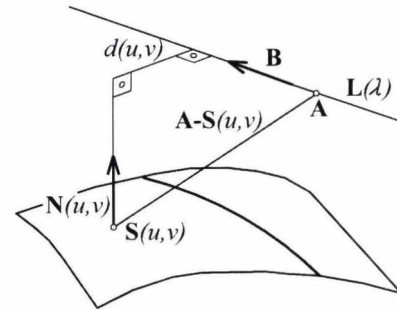


Figure 2: Distance between surface normal and the lightsource

The perpendicular distance $d(u, v)$ between the normal $N(u, v)$ at a surface point $S(u, v)$ and the linear light source is:

$$d(u, v) = \frac{|[B \times N(u, v)] \cdot [A - S(u, v)]|}{|B \times N(u, v)|}.$$

To obtain points on highlights $d(u, v) = 0$ must be solved for the control points of $S(u, v)$ which is a highly complex task. Even more complex is the inverse task, to determine surface parameters corresponding to prescribed shape of highlights. We solved this by applying genetic algorithm.

A more complex and detailed overview of the surface quality can be obtained by a series of highlight lines. This is realized by several lightsources, parallel to each other. During the current research, we developed, and used a robust, computation method with high accuracy for creating highlight lines. Details of the method can be found in ³ and not discussed here.

3. Concept of genetic algorithm

Genetic algorithms were first used by J. H. Holland ⁵. The basic idea is to try to mimic a simple picture of natural selection in order to find a good solution. Genetic algorithms work on a collection of solutions, population of individuals. The individuals are coded by chromosomes, composed of genes that contain the parameters of the solution to be optimized. Binary coding is frequently used, however for problems with continuous variables, a real coded genetic algorithm (RCGA) is more efficient ². Set of all possible chromosomes form the search space, where the genetic algorithm tries to find the best chromosome, corresponding to the best solution.

Genetic search starts with an initial population, which is selected randomly from the search space. In each generation individuals are evaluated according to predefined criteria called fitness function. Fitness quantifies the optimality of a solution so that that particular solution may be ranked against all the other solutions. Fitness value of an individual correlates closely with the goal of search, and must be computed quickly.

As the result of the selection procedure, individuals are chosen for breeding. Part of the selection step can be the optional "elitism" strategy, where the best chromosomes (as determined from their fitness evaluations) are placed directly into the next generation. This guarantees the preservation of the best chromosomes at each generation.

New individuals created into the population by genetic operators. Most frequently applied operators are selection, crossover, and mutation. The objective of crossover is to generate a new individual by inheritance. It combines the

```

Program GA

g:=1 { generation counter }
P(g) Initialization of the first population
P(g) Population evaluating

Cycle until (stop condition) or (g = gmax)
  g:=g+1
  P(g) scaling P(g-1)
  P(g) selection from P(g-1)
  P(g) crossover
  P(g) mutation
  P(g) population evaluating
Cycle END

Program END

```

Figure 3: Pseudo Code of a Genetic Algorithm

features of two (or more) chromosomes to form one or more offspring, with the possibility that combination of good chromosomes may generate better ones.

The mutation operator arbitrarily alters one or more components, genes of a selected chromosome so as to increase the structural variability of the population. The role of mutation in GAs is that of restoring lost or unexplored chromosomes into the population to prevent the premature convergence of GA to suboptimal solutions. Each position of every chromosome in the population undergoes a random change according to a probability defined by the mutation rate. The algorithm runs while a stopping condition is met. Stopping condition is associated with an acceptable fitness value.

In most cases parameters of a good algorithm are varying depending on the problem type. For a good, and effective GA, operator types and their parameters should be charily selected and tested. Fitness function also must be carefully composed and fine-tuned.

4. GA for highlight line correction

The goal of the genetic search is to modify the shape of the surface in order to obtain a better shape and distribution of highlight lines than existing one. General principles, methods, or even downloadable algorithms are available for GAs. Still, to solve complex technical problems, specific solutions for genetic coding, genetic operators, selection mechanisms, fitness specifications are needed within the general framework of GAs. Subsequently we discuss particular issues of applying GA to correct of surface through highlight lines.

4.1. Representation of genes

To modify the surface one or more surface parameters must be changed, and an acceptable value for them must be found. Free form surfaces as described in chapter 2, are determined by a number of parameters. However, the most effective parameter for surface modification is the control point. Using them, we can create a simple structured gene which is easy to handle, and which influences the surface shape directly and influentially.

The control points that have influence on the surface area to be modified can be computed from the basis functions corresponding to a control point. Integrating the basis functions over the area of interest a strength of influence parameter for each control points b_{ij} can be obtained. It plays important role when determining the size of the search space. The control points to be modified are sequentially represented in the genetic code in the genes. A gene g_γ consist of ΔP_{ij} and b_{ij} :

$$g_\gamma = \Delta P_{ij}(x, y, z, b_{ij}).$$

During the search, genetic operators are applied only on control points. The chromosome of a surface to be modified is composed of genes of all control points:

$$c_\beta = (g_1 \dots g_J \dots g_J),$$

where c_β is the chromosome of a surface in the population, and J is the number of control points to be modified. The modified surface is described by the equation:

$$S(u, v) = \sum_{i=0}^m \sum_{j=0}^n (P_{ij} + \Delta P_{ij}) N_{ik}(u) N_{jl}(v).$$

4.2. Calculation of fitness

Fitness function contains geometric deviation information between actual and desired highlight lines, and it consists of two components. These are the accuracy, and the shape similarity. Accuracy is based on the distance, while shape similarity on tangency error between actual and desired highlight lines. In fitness function they are present as the sum of average variation of a highlight lines. Denote by h_i^d a highlight line, smoothed by the designer, and h_i^c the corresponding highlight curves, created during genetic search. Curves are evaluated at several parameter values for comparison. Then the distance error component of the fitness function is

$$f_d = \sum_{i=1}^l \left(\sum_{k=1}^{n_i} \left(d_i(t_k) - \frac{1}{n_i} \cdot \sum_{k=1}^{n_i} d_i(t_k) \right)^2 \cdot \frac{1}{n_i} \right),$$

where $d_i(t_k) = |h_i^c(t_k) - h_i^d(t_k)|$

n_i – number of curve points where the evaluation was carried out for curve h_i^c and h_i^d

l – the number of inspected highlight lines.

The tangency error f_i is

$$f_i = \sum_{i=1}^l \left(\sum_{k=1}^{n_i} \left(a_i(t_k) - \frac{1}{n_i} \cdot \sum_{k=1}^{n_i} a_i(t_k) \right)^2 \cdot \frac{1}{n_i} \right),$$

where $a_i(t_k) = \text{arc}(h_i^c(t_k)) - \text{arc}(h_i^d(t_k))$

$h_i^c(t_k)$ and $h_i^d(t_k)$ denote tangent vectors at t_k .

The ratio of the two components of fitness may vary during the GA process. Details of the strategy for the adjustment of fitness components are explained in chapter 5.2.

4.3. Scaling and selection methods

Selection is carried out using scaled fitness proportional selection⁴ (also known as roulette wheel selection).

The fitness of chromosomes are scaled before selection process. Scaling is used, to promote the better exploration of search space, and maintain population's diversity. There are a number of available scaling algorithms, we analysed those, which promise the largest exploration. Scaling algorithms in first step sort fitness values. The fitness values are then replaced by their ordinal numbers, and scaled. Scaling relations are summarized in Table 1.

Basic scaling

At this scale type², the fitness is simply replaced with the ordinal number β . This scaling increases the value inequalities, in a sharp way. We can expect that this scaling has good effect on genetic search in its latter phase. When chromosome variance becomes low, such sharp scaling may promote distinction between chromosomes.

Rank scaling

At this scaling type, fitness is distributed along a second order function, which unlike basic scaling will equalize the best chromosomes, and cut the worst off⁴. This type is expected to work well at the beginning of the genetic search.

Linear scaling

This scaling is capable of adopting itself to the different stages of genetic search². This function is realized through variable p . If p is close to 0 it promotes the survival of the best chromosomes. Close to 1 value will result in equal chances for every chromosome.

Scaling type	Relation
basic	$f' = \beta$
rank	$f' = \frac{1}{\sqrt{\beta}}$
linear	$f' = p - \frac{2(\beta-1)(p-1)}{\beta_{\max}-1}$

Table 1: Relations of fitness scaling

4.4. Types of crossover

On selected parent chromosomes different types of crossover^{2,4} were investigated.

Denote $c_\beta^1 = (g_1^1 \dots g_\gamma^1 \dots g_J^1)$ and $c_\beta^2 = (g_1^2 \dots g_\gamma^2 \dots g_J^2)$ the two parent chromosomes to which crossover operation is to be applied, and the offspring pool $O_\kappa = (o_1^\kappa, \dots, o_\beta^\kappa, \dots, o_{\Gamma/2}^\kappa)$. Where Γ is the number of chromosomes in the population, κ is the index of the offspring.

Arithmetic crossover

Two offspring is produced. $O_\kappa = (o_1^\kappa, \dots, o_\beta^\kappa, \dots, o_{\Gamma/2}^\kappa)$, $\kappa = 1, 2$ where $o_\beta^1 = \vartheta g_\gamma^1 + (1 - \vartheta) g_\gamma^2$ and $o_\beta^2 = g_\gamma^2 + (1 - \vartheta) g_\gamma^1$. Parameter ϑ determines the offspring's position between parents. We chose to use $\vartheta = 0.8$.

BLX- α crossover

This type of crossover produces one offspring. $O = (o_1, \dots, o_\beta, \dots, o_\Gamma)$ where o_β is chosen from the interval $[c_{\min} - I \cdot \alpha, c_{\max} + I \cdot \alpha]$ randomly. Where $c_{\max} = \max(c_\beta^1, c_\beta^2)$, $c_{\min} = \min(c_\beta^1, c_\beta^2)$, $I = c_{\max} - c_{\min}$. Value of α extends offspring's position over the domain, defined by the parents and determines the action interval of a gene⁴. We used $\alpha = 0.2$.

BLX- α - θ crossover

It is similar to BLX- α except it uses yet another constant θ for extending action interval and which introduces asymmetry. The offspring o_i is chosen from interval $[g_{\min} - I \cdot \alpha, g_{\max} + I \cdot \alpha \cdot \theta]$.

Wright's heuristic crossover

This crossover uses population's best chromosome as first parent. One offspring is created: $o_\beta = \rho \cdot (c_\beta^1 - c_\beta^2) + c_\beta^1$, where ρ is a number randomly chosen from the interval $[0, 1]$.

4.5. Stop condition

Stop condition is relied on the allowable residual error, which is computed by the comparison of the original and the redesigned highlight lines.

Let μ_{stop} denote the stop condition

$$\mu_{stop} = \begin{cases} \text{true} & \text{if } f_t^\tau / f_t^0 \leq \varepsilon \text{ or } \tau = \tau_{\max} \\ \text{false} & \text{otherwise} \end{cases}$$

where ε is the allowable tangency error

f_t^τ tangency error of best chromosome at generation number τ .

In case the search fails to meet the stop condition it is stopped at the maximum number of generations τ_{\max} . During the analysis, we used the following stop conditions:

$$\varepsilon = 0.1 - \tau_{\max} = 100$$

5. Application and results

We applied our method to a number of real world examples (Table 2). Data for the example surfaces are given in the Table 2. Shape of the surfaces, defective and corrected highlight lines are shown in the Appendix.

ID	Name	Max. error [mm]	Total number of CPs	Number of affected CPs
S1	Turbine shovel	0.09	600	72
S2	Z3 fender	0.41	600	29
S3	Cam. trunk lid	0.86	924	61
S4	FIAT fender	7.21	156	24
S5	A6 trunk lid	4.51	874	55
S6	Freeform design	1.28	1152	42

Table 2: Analysed real world surfaces

First we determine the reflection characteristic of the surface and find areas where correction is needed, that is - find the errors. For thorough examination, several different light-source settings have to be used. These settings are implemented by changing two parameters: lightsource density and mutual position with the surface.

Figure 4a displays the highlight lines of a freeform surface. It can be easily discovered, that it has irregularities around the central region (area indicated by circle). After the defective highlight lines were spotted, correction takes place. In Figure 4b highlight lines before and after correction are shown

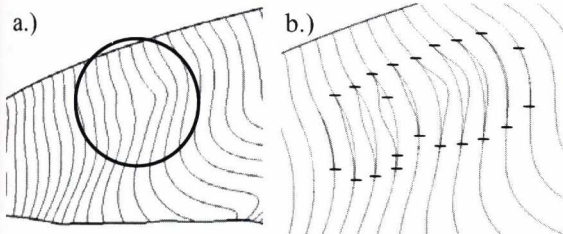


Figure 4: a.) Highlight lines of a freeform surface
b.) correction of defective highlight lines (surface S2)

5.1. Initial population

The members of the initial population are randomly changed chromosomes of the original surface.

$$c_{\beta}^1 = \begin{cases} c_{\beta}^0 + r_{\beta} & \text{if } 0.5 \leq \rho < 1 \\ c_{\beta}^0 - r_{\beta} & \text{if } 0 < \rho < 0.5 \end{cases}$$

where c_{β}^0 is a chromosome of the initial surface, c_{β}^1 is a chromosome of the initial population and ρ is a number,

randomly chosen from the interval [0,1]. The size of the population kept on constant value $\Gamma=50$ during the search.

5.2. Evaluation of different fitness types

Distance error is proved to assist genetic search in exploring areas of search space that ensure accurate highlight lines. In the other hand however, it hasn't helped to improve shape similarity. Tangency error behaves in opposite way: it helps producing better shape similarity of highlight lines, but on the expenses of their accuracy. We could eliminate the drawbacks if we let the distance dominate in the beginning of the search and make the tangency error come into effect at the end. To realize this idea, surface fitness as a weighted sum of distance and the tangency error is needed. Let f be the fitness of the surface, then:

$$f = w_d \cdot f_d + w_t \cdot f_t,$$

where w_d - weight of distance error
 w_t - weight of tangency error

The influence of distance and the tangency error is studied through different compositions. To make easier distinction between different types of fitness we use abbreviations.

- F1 - contains only distance variation error.
- F2 - contains only tangency variation error.
- F3 - In the beginning the ratio of distance / tangency error weights is 75 / 25 %. The ratio is turning over as linear function, connected to the maximal number of generations.
- F4 - In the beginning the ratio of distance / tangency error weights is 75 / 25 %. The ratio is turning over as linear function, connected to the variance of chromosomes.
- F5 - In the beginning the ratio of distance / tangency error weights is 75 / 25 %. The ratio turns over, when variation of chromosomes falls below 15% of the original.
- F6 - In the beginning the ratio of distance / tangency error weights is 100 / 0 %. The ratio turns over, when variation of chromosomes falls below 15% of the original.

Distance and tangency errors have different measures, dimensions and range of change. To make them suitable for mutual appearance in fitness, they are normalized as

$$f'_d = f_d^{\delta} \cdot 1 / f_d^0, \quad f'_t = f_t^{\delta} \cdot 1 / f_t^0,$$

where f_d^{δ} and f_t^{δ} are the error components of the fitness function of corrected highlight lines, f_d^0 and f_t^0 are the error components of the fitness function of original highlight lines. Thus, the fitness function can be written in following form:

$$f = w_d(c_{\text{var}}, \tau) \cdot f'_d + w_t(c_{\text{var}}, \tau) \cdot f'_t.$$

We run GA for each surface and for each fitness type, and the stop condition was reached at every fitness type. It Table 3, ranking of fitness types is shown

F1	F2	F3	F4	F5	F6
5.25	3.12	3.62	2.25	3.37	3.37

Table 3: Average performance order of fitness types

while Figure 5 gives an overview of their influence on genetic search performance. Compared to other types, we achieved only medium results with F1. Stop condition was almost every time reached as last, and after taking a closer look at the results we learned that the shape similarity is barely satisfactory.

When F2 was used, results were reached much earlier than with F1. The shape similarity was satisfactory but the distances between resulting and desired curves were larger than at F1. With F3 the results showed improvement both from aspect of distance, and tangency error, however the convergence was slower than with F1 and F2. Another drawback of this method was the fixed number of generations: it can make the search groundlessly long.

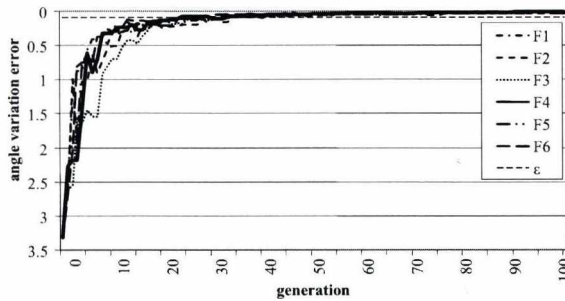


Figure 5: Development of tangency error at different fitness functions (surface S4)

To make the genetic search more effective, at F4, F5 and F6 the change of ratio was set to the chromosome variability change. As can be seen in Table 3, types F5 and F6 reached the stop condition roughly after the same generation number. A real surprise was the performance of F4. In half of the cases, it had the quickest convergence, and it never performed as last.

5.3. Evaluation of scaling types

According to our experience, the order of performance of scaling types was unambiguous: basic selection type brought the best results in all of the cases. This might be attributed to its property to promote exploration of search space. This can be a balance to the susceptibility of RCGAs to premature convergence⁴. An example for the behavior of different scaling types is shown in Figure 6.

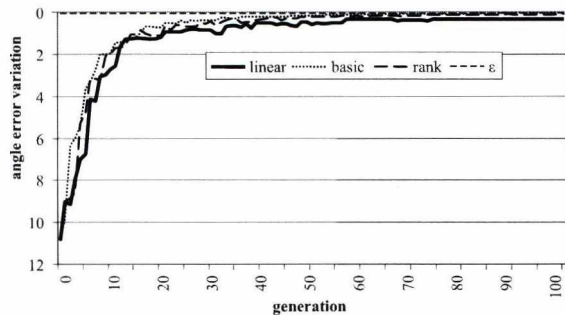


Figure 6: Development of tangency error at different scaling types (surface S3)

5.4. Evaluation of crossover types

The aim of crossover analysis was to choose the best method for offspring creation and the exploration/exploitation ratio. During the analysis of BLX- α , number of α value was applied. The gained search results showed that values near 0.2 promise the quickest results. This value puts the genes into interval of relaxed exploitation. The extension of action interval is symmetrical. At crossover type BLX- α - θ the extension is larger and the action interval is asymmetric. This shifts the middle of the action interval. In case of an appropriate shift, it can reduce the probability of premature convergence and increases the exploration. During the experiments, $\theta = 0.4$ brought the best results. Comparing with the other types of crossovers, in most of the cases BLX- α - θ was the quickest of all (Table 4).

BLX- α	BLX- α - θ	Uniform Arithmetic	Wright Heuristic
2.12	1.75	3.12	3

Table 4: Average performance order of crossover types

The uniform arithmetic and the Wright Heuristic crossovers were mainly in the middle of the performance comparison list. In Figure 7 the performance of tested crossover types can be tracked.

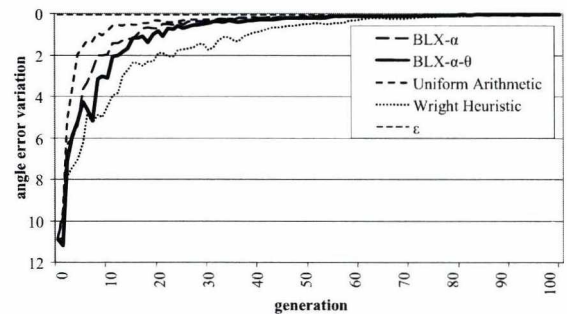


Figure 7: Development of tangency error at different crossover types (surface S1)

6. Conclusion and future work

Our method for improving the quality of freeform surfaces by modifying its highlight lines is capable of finding the surface that corresponds to desired highlight lines with prescribed precision. Best performing genetic operators, strategies and parameters of the genetic algorithm was determined. These are given in Table 5.

Fitness	Scaling	Crossover	Mutation
F4	basic	BLX- α - θ $\alpha = 0.2, \theta = 0.4$	Non uniform $q=0.8$

Table 5: Operators of best performing genetic algorithm

The method has advantages over existing ones. Comparing to other methods it can find the solution regardless of surface representation, complexity of surface shape and highlight lines. Our method is robust and intuitive, because control point modification is achieved through genetic algorithm, without computing highly nonlinear correlation

between control points and highlight lines. It allows the designer to correct surface by simple redesigning the abnormal portions of the highlight lines.

References

1. K.-P. Beier and Y. Chen. Highlight-line algorithm for real time surface-quality assessment, *Computer-Aided Design*, **26**(4):268-277, 1994.
2. K. Deb, A. Anand, D. Joshi. A computationally efficient evolutionary algorithm for real-parameter optimization, *Evolutionary Computation*, **10**(4), 2002.
3. Gy. Gyurecz and G. Renner. Robust computation of reflection lines, *Journal of Machine Manufacturing*, **49**(E6): 65-68, 2009
4. F. Herrera, M. Lozano, J. L. Verdegay. Tackling real-coded genetic algorithms, *Artificial Intelligence Review*, **12**(4):265-319, 1998.
5. J. H. Holland. *Adaption in natural and artificial systems*, MIT Press Edition, 1998.
6. E. Kaufmann and R. Klass. Smoothing surfaces using reflection lines for families of splines, *Computer-Aided Design*, **20**(6):312-316, 1988.
7. C. Zhang and F. Cheng. Removing local irregularities of NURBS surfaces by modifying highlight lines, *Computer-Aided Design*, **30**(12):923-930, 1998.

Appendix

The appendix present the results in visual mode. Pictures show the surfaces in following order:

- Rendered image
- Highlight lines of the original surface
- Highlight lines of the corrected surface

Please notice that while rendered image seems to be smooth, highlight lines disclose irregularities.

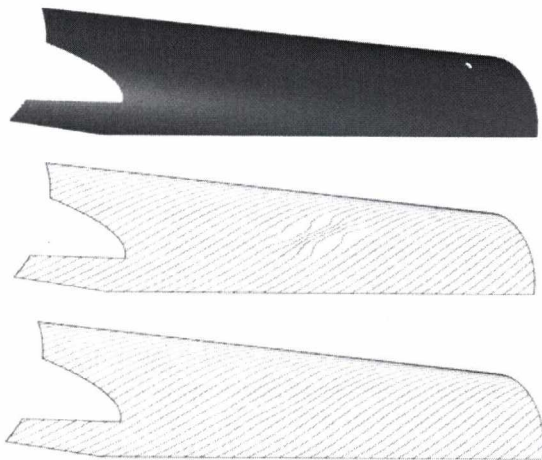


Figure 8: Surface S1 in its original and corrected form



Figure 9: Surface S2 in its original and corrected form



Figure 10: Surface S3 in its original and corrected form

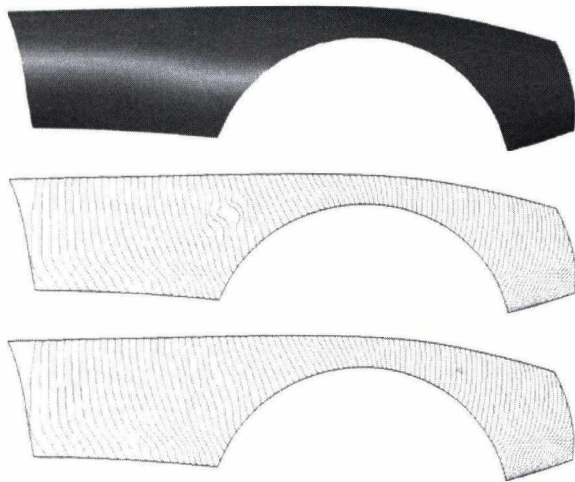


Figure 11: *Surface S4 in its original and corrected form*



Figure 12: *Surface S5 in its original and corrected form*

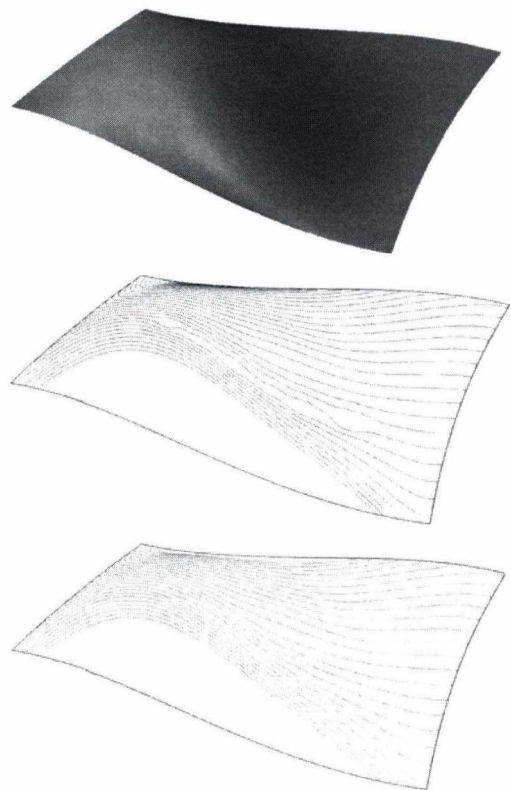


Figure 13: *Surface S6 in its original and corrected form*

Approximating non-metrical distances in 3D

András Hajdu and Tamás Tóth

¹ Faculty of Informatics, University of Debrecen, 4010 Debrecen, POB 12, Hungary

Abstract

In this paper we give an approach for approximating the 3D non-metrical Minkowski distances with digital distance measurement tools. We also present a chamfering algorithm for fast computation. The distance measurement is based on weighted neighborhoods and the optimal weights are found by comparing the corresponding spheres of the distance functions.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modelling

1. Introduction

The most usual distance measurement tool in digital image processing and in discrete geometry is the Minkowski one, which is also known as the L_p one. The general formula of L_p in $n \in \mathbb{N}$ dimension is as follows:

$$L_p(q, r) = \begin{cases} \left(\sum_{i=1}^n |q_i - r_i|^p \right)^{1/p}, & \text{for } 0 < p < \infty, \\ \max_{i=1}^n (|q_i - r_i|), & \text{for } p = \infty, \end{cases}$$

$$q, r \in \mathbb{R}^n.$$

The L_2 distance is the Euclidean one, and L_1 is denoted as Neumann or city-block distance, while the L_∞ as Moore or chessboard distance. It is also known that the general L_p , $1 \leq p \leq \infty$ distance measurement function is a metric, that is the following conditions hold:

- $\forall q, r \in \mathbb{R}^n : L_p(q, r) \geq 0$, $L_p(q, r) = 0 \Leftrightarrow q = r$ (positive definite),
- $\forall q, r \in \mathbb{R}^n : L_p(q, r) = L_p(r, q)$ (symmetric),
- $\forall q, r, s \in \mathbb{R}^n : L_p(q, r) + L_p(r, s) \geq L_p(q, s)$ (has the triangle inequality).

If $p < 1$, the third condition fails. For more details see ^{11, 15}.

The purpose of this paper is to find an approximation of L_p for $0 \leq p \leq 1$ in 3D. This investigation is a natural continuation of our former work for 2D ¹¹. We also aim

to present a quick tool to implement the approximation in the digital domain. Such an effective algorithm is chamfering ¹, which can process a distance map processing an input set using neighborhood functions. First, we have to describe the geometric structure which is used for approximating L_p , and we also have to describe the mathematic properties of the approximation.

2. Overview of the approximation

The approximation of L_p is performed in a geometrical basis. That is, we approximate the unit sphere of L_p , $0 \leq p \leq 1$ with a *star-shaped* object shown in Figure 1 and 4. The approximating object is selected to be easily generatable by local neighborhoods to be able to build quick distance measurement upon it with a fair approximating performance (see Figure 2).

The boundary of the approximating object in the octants can be defined by planes defined by 3-3 keypoints each. In the positive octant these points are the followings: $P_1(1, 0, 0)$, $P_2(0, 1, 0)$, $P_3(0, 0, 1)$, $P_4(x_0, x_0, 0)$, $P_5(x_0, 0, x_0)$, $P_6(0, x_0, x_0)$. From these points the following planes can be defined: $R_1(P_1, P_4, P_5)$, $R_2(P_2, P_4, P_6)$, $R_3(P_3, P_5, P_6)$. The corresponding keypoints in the other octants can be gained by changing the signs of the coordinates. However, symmetry makes it sufficient to investigate the positive octant only.

The three planes intersect at the point $X_1(x_1, x_1, x_1)$. The intersecting point of two planes and the plane defined by x and y axes is $X_0(x_0, x_0, 0)$. The approximation can be improved further, if we make X_1 independent of the intersect-

ing point of the planes, see Figure 3. By this improvement we can divide the investigation to *special* and *general* types. That is, our main goal is to find x_0 and x_1 .

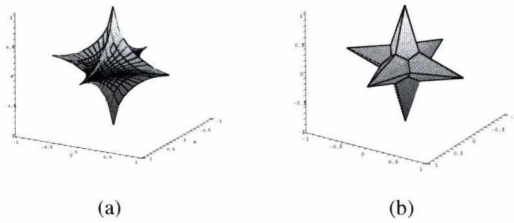


Figure 1: The L_p sphere with $p = 6/10$ (a), and its approximation with the star, where $m = -1/3$, (b).

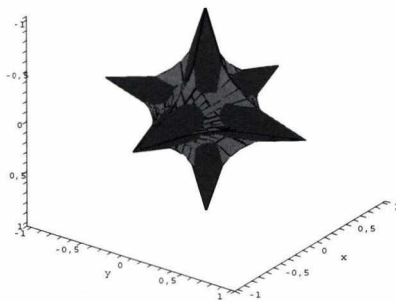


Figure 2: Approximating the L_p , $p = 6/10$ sphere with the approximating object $m = -1/3$.

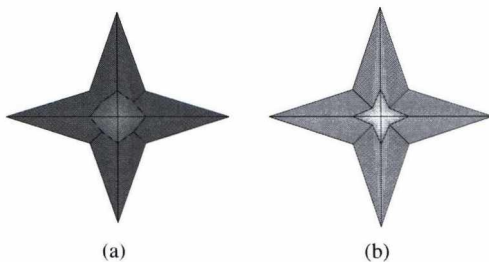


Figure 3: Different types of the approximating objects. The difference is in the number of the defining planes, and thus the coordinates of x_1 . The special case (a) is constructed by 24 planes, while the general case (b) is constructed of 48 planes. For the corresponding selection of x_1 see Section 5.2.

3. Distance functions for approximation

We will use neighborhood sequences to define the family of discrete distance functions for approximating L_p , $0 \leq p \leq 1$ distances. A neighborhood is a pair (P, w) , where the point set $P \subseteq \mathbb{Z}^3$ is finite, and $w : P \rightarrow \mathbb{R}_{\geq 0}$ is a so-called weight function on P , see ¹⁰. For $p \in P$, $w(p)$ is the weight of p . Let Λ be a finite set of neighborhoods. A 3D-neighborhood sequence is defined as a sequence $N = (N_i)_{i=1}^{\infty}$ over Λ , that is, $N_i \in \Lambda$ for all $i \in \mathbb{N}$. Let S_3 denote the set of all 3D-neighborhood sequences. If for some $j \in \mathbb{N}$, $N_i = N_{i+j}$ for all $i \in \mathbb{N}$, then N is called periodic with period j . In this case we use the brief notation $N = \overline{N_1 N_2 \dots N_j}$. If $j = 1$ then N is called a constant neighborhood sequence.

We can measure distance by the help of neighborhood sequences in a natural way (see e.g. ^{3, 7, 13, 18}). Let q and r be two points in \mathbb{Z}^3 , and $N = (N_i)_{i=1}^{\infty} \in S_3$, with $N_i = (P_i, w_i)$. The point sequence $S = [q_0, q_1, \dots, q_m]$, where $q = q_0$, $r = q_m$, and $q_i - q_{i-1} \in P_i$, is called an N -path between q and r . The length of S is defined as $\sum_{i=0}^{m-1} w_i(q_{i+1} - q_i)$. The N -distance $w(q, r; N)$ between q and r is defined as the length of a shortest N -path between them, if such a path exists. The distance function generated by N is denoted by $d(N)$. For more results on the various families of neighborhood sequences, especially on their approximation power, consult with ^{1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 14, 15}. In our current investigations we deal only with constant neighborhood sequences, consisting of symmetric N_6 -neighborhoods, see Figure 5. As an approximating function, we will consider the distance function:

$$D = \min(d(NX), d(NY), d(NZ)), \quad (1)$$

where for the neighborhood sequences $NX = \overline{N_x}$, $NY = \overline{N_y}$, and $NZ = \overline{N_z}$, with $N_x = (P_x, w_x)$,

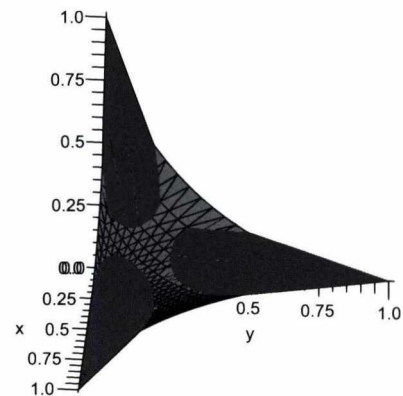


Figure 4: Basic overlapping case demonstrating special type of approximation.

$N_y = (P_y, w_y)$, $N_z = (P_z, w_z)$, where $P_x = P_y = P_z = \{(1, 0, 0), (-1, 0, 0), (0, 1, 0), (0, -1, 0), (0, 0, 1), (0, 0, -1)\}$, and we introduce WX , WY and WZ for the weights of a step into the direction of x , y and z respectively, see Figure 5.

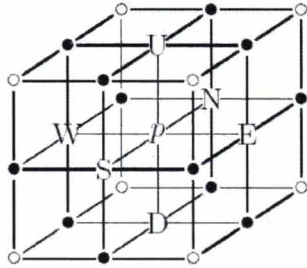


Figure 5: The neighborhood of a point p in 3D. $WX = W = E$, $WY = N = S$ and $WZ = U = D$, respectively.

The disc of \mathcal{D} will be referred shortly as a star and denoted by $B_{\mathcal{D}}$. Our aim is to find optimal WX , WY and WZ values which case the corresponding star approximates the best the L_p sphere for $0 \leq p \leq 1$. Note that taking the minimum to define \mathcal{D} is the key step to extend the former approaches e.g. existing for L_3 with $1 \leq p \leq 1$.

3.1. Chamfering

It is a natural requirement to obtain distance maps rapidly, thus, several algorithms were proposed for this aim. One of them is chamfering¹, which calculates distance maps using weighted 6-, 18- or 26-neighborhoods in 3D, such that the chamfer distance of two points is just the length of any 26-connected path between them having the minimal sum of weights.

Chamfer distance transformations are performed in two (forward/backward) scans over the source using weighted neighborhoods.

In the forward step, the procedure starts from the one corner of the source, moves from left to right and from top to bottom, using the forward mask. In the backward step this is reversed.

To find a 2-pass chamfering algorithm for our purposes, we recall the idea (1) of taking the minimum of more distance functions. Namely, we calculate more distance maps simultaneously, based on the weighted neighborhoods. The number of the maps depends on type of the approximation: 6 maps for the general case and 3 for the special one. Then we derive the distance map of \mathcal{D} as the minimum of the other maps. Since the temporary distance maps can be calculated within the same 2-pass chamfering procedure, and final one in its backward step, our algorithm belongs to the family of 2-pass chamfering algorithms. For more details see^{16, 17}.

4. Spheres of distance functions

In this section, we describe the geometrical properties of the L_p sphere, and of the approximating star object, such as volume, surface and the special points like x_0 and x_1 .

4.1. Geometry of the L_p sphere

We have to describe the properties of the L_p sphere and the approximating object, such as volume and surface. After having these properties of the unit L_p sphere, we can describe the approximating object for any $0 \leq p \leq 1$.

4.1.1. Variable substitution

The V volume and S surface of the L_p sphere in the positive octant can be calculated as follows:

$$V = \int_{x=0}^1 \int_{y=0}^{(1-x^p)^{1/p}} (1-x^p+y^p)^{1/p} dx dy, (2)$$

$$S = \int_{x=0}^1 \int_{y=0}^{(1-x^p)^{1/p}} \left(1 + \frac{\partial f(x,y)}{\partial x} + \frac{\partial f(x,y)}{\partial y} \right)^{1/2} dx dy, (3)$$

$$\text{where } f(x,y) = (1-x^p+y^p)^{1/p}. (4)$$

Unfortunately the complexity of the formulae does not allow us to compute the surface and volume for arbitrary p . A solution for that is a coordinate transformation from (x,y) to (z,φ) as follows:

$$\begin{aligned} z &= (x^p + y^p)^{1/p}, \\ \varphi &= \arctan\left(\frac{y}{x}\right)^{p/2}, \quad 0 \leq \varphi \leq \frac{\pi}{2}. \end{aligned} (5)$$

Then the Jacobi matrix and its determinant for the coordinate transformation can be calculated as:

$$J = \begin{bmatrix} \partial x / \partial z & \partial x / \partial \varphi \\ \partial y / \partial z & \partial y / \partial \varphi \end{bmatrix} (6)$$

$$= \begin{bmatrix} \cos(\varphi)^{2/p} & \sin(\varphi)^{2/p} \\ -\frac{2z \cos(\varphi)^{(2/p)} \sin(\varphi)}{p \cos(\varphi)} & \frac{2z \sin(\varphi)^{(2/p)} \cos(\varphi)}{p \sin(\varphi)} \end{bmatrix}, (7)$$

$$JD = \frac{2z \cos(\varphi)^{(2/p)} \sin(\varphi)^{(2/p)} (\cos(\varphi)^2 + \sin(\varphi)^2)}{p \sin(\varphi) \cos(\varphi)}. (8)$$

4.1.2. The volume of L_p sphere

The variable substitution gives us $f(x,y) = (1-z^p)^{1/p}$. Now, based on (2), we have:

$$\begin{aligned}
 V_{L_p} &= \int_{z=0}^1 \int_{\varphi=0}^{\pi/2} f \text{JD} \, dz d\varphi \\
 &= \frac{2}{p} \int_{z=0}^1 (1-z^p)^{(1/p)} z \, dz \\
 &\quad \int_{\varphi=0}^{\pi/2} \cos(\varphi)^{-\left(\frac{p-2}{p}\right)} \sin(\varphi)^{-\left(\frac{p-2}{p}\right)} d\varphi. \quad (9)
 \end{aligned}$$

4.1.3. The surface of L_p sphere

Computing the surface is similarly executed to the computation of the volume. That is:

$$\begin{aligned}
 S_{L_p} &= \int_{z=0}^1 \int_{\varphi=0}^{\pi/2} f \text{JD} \, dz d\varphi \quad (10) \\
 &= \int_{z=0}^1 \int_{\varphi=0}^{\pi/2} \sqrt{1 + \left(\frac{\partial L_p}{\partial x}\right)^2 + \left(\frac{\partial L_p}{\partial y}\right)^2} \text{JD} \, dz d\varphi.
 \end{aligned}$$

4.2. Geometry of the approximating object

The most simple case is if the object is defined by three planes. We take three points to define one of these planes. Because of symmetry, the missing points to define the other planes:

$$\begin{aligned}
 &P_1(x_0, 0, x_0), \quad P_2(x_0, x_0, 0), \quad P_3(1, 0, 0), \\
 &P'_1(0, x_0, x_0), \quad P'_2(x_0, x_0, 0), \quad P'_3(0, 1, 0), \\
 &P''_1(x_0, 0, x_0), \quad P''_2(0, x_0, x_0), \quad P''_3(0, 0, 1).
 \end{aligned}$$

We use the following points in our further computations:

$$A(x_0, 0, x_0), \quad B(x_1, x_1, x_1), \quad C(1, 0, 0). \quad (11)$$

4.2.1. Surface

In the positive octant, the S surface of the approximating object is the union of three deltoids. The A, B, C points defined as special type of approximation divide the deltoid into two symmetric parts along its longer diameter. We have to take the triangle from A, B, C , take its area, and multiply with 6 for the whole surface. Thus the formula for surface is independent of x_1 . That is, the formula for the special type can be applied for the general type, as well. The area of the triangle can be computed e.g. by Heron's formula:

$$\begin{aligned}
 a &= (3x_1^2 - 4x_0x_1 + 2x_0^2)^{(1/2)}, \\
 b &= (3x_1^2 - 2x_1 + 1)^{(1/2)}, \\
 c &= (2x_0^2 - 2x_0 + 1)^{(1/2)}, \\
 s &= \frac{a + b + c}{2}, \\
 S_{\text{obj}} &= 6\sqrt{s(s-a)(s-b)(s-c)}. \quad (12)
 \end{aligned}$$

4.2.2. Volume

By computing the volume, the general type is investigated. The object considered in special type can be divided by a plane which is defined by the axis at its node and by the line $x = y = z$. The plane described above and the object have two intersecting points B and C . Moreover, we have to consider the origin $D(0, 0, 0)$, and the point A , too. That is, the object, whose volume is to be computed is a tetrahedron. The base triangle is composed by points A, C, D , and its height is x_1 . The area of the triangle can be computed easily, because the length of the longest side is 1, and its height is x_1 , see (13).

$$V_{\text{obj}} = 6 \frac{T_b h}{3} = 6 \frac{(x_0/2)x_1}{3} = x_0 x_1. \quad (13)$$

5. Approximating the L_p sphere

We can apply two kinds of approximation for the L_p sphere: "fully covering case" and "overlapping case". In the former case the z value of the approximating object is greater than or equals to the L_p one for all the corresponding (x, y) values. In other words, the star-object covers the L_p sphere. In the overlapping case the star-object and the L_p sphere can overlap.

5.1. Fully covering case

5.1.1. Special type

The simplest case is the natural extension of the 2D case. We intersect the object and the L_p sphere by the plane defined by the x and y axes. Thus, we have the 2D case. From [11] we have:

$$x_0 = 2^{(-1/p)}.$$

In the special case X_1 is the intersection point of three planes in the octant:

$$x_{1s} = -\frac{x_0}{x_0 - 2}.$$

As L_p is concave, we only have to investigate the function in some special points. These are $X_0(x_0, x_0, 0)$ and $X_1(x_1, x_1, x_1)$, respectively. Since we have chosen X_0 as the extension of the 2D approximation, we have to check the relation of the L_p and approximating values in $(x_1, x_1, 0)$. In Figure 6, we present the L_p and the approximation values for several p values. It can be seen, that $x_1 \geq L_p(x_1, x_1)$ for every p .

To analyse the correctness of the approximation, we take the slice of the approximation on the plane defined by the triangle $[(x_0, x_0, x_0), (x_0, 0, x_0), (x_0, x_0, 0)]$. In this 2D case the radius of the sphere and the object is x_0 and the slope of the approximating line is $m = -1$. It is trivial that this approximation is not optimal in 3D. We try to refine our approximation.

5.1.2. General type

It can be a simple improvement to add X_1 as a common point of the approximating star-object and L_p . In this case, the approximation objects are triangles, whose nodes are on the L_p sphere. Since L_p is concave, this approximation always leads to the covering case. By solving the following equations:

$$z = x = y = (1 - x^p - y^p)^{(1/p)} = (1 - 2z^p)^{(1/p)},$$

we have (14) for x_{1g} :

$$x_{1g} = 3^{(-1/p)}. \tag{14}$$

5.1.3. Results

Table 1 summarizes the approximation results of the covering case regarding symmetric difference.

5.2. General overlapping approximation

5.2.1. Choosing the value of x_1

We need a criteria to measure the accuracy of the approximation. We have chosen the compactness ratio defined as follows:

$$CR = \frac{\text{volume}}{\text{area}^2}.$$

The main tasks in general overlapping approximation are to choose the proper x_0 and x_1 points. We have four ideas for choosing x_1 . The common expectation is that the compactness ratio of the approximating object and L_p must coincide ($CR_{L_p} = CR_{cs}$).

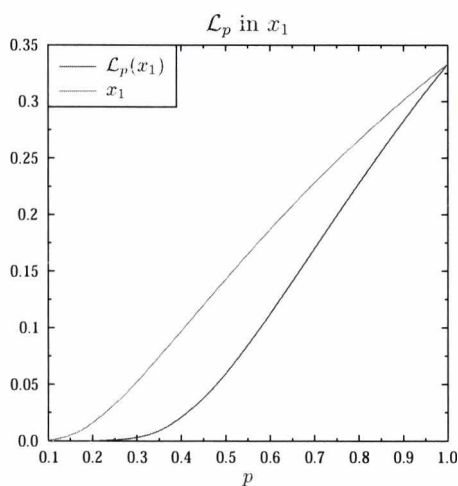


Figure 6: x_1 and L_p values in $(x_1, x_1, 0)$, $p \in [0, 1]$.

Case #1 The simple extension of the 2D case, X_1 is the intersecting point of the three planes of the star-object. Also noted as special type of approximation.

$$x_{1a} = \frac{x_{0a}}{x_{0a} - 2}$$

Case #2 The slope m of 3D star and the slope m' of the refinement in 2D are the same.

$$x_{1b} = \frac{x_{0b}^2}{x_{0b}^2 - x_{0b} + 1}$$

Case #3 The X_1 is on the L_p sphere as in the general covering case.

$$x_{1c} = 3^{-1/p}$$

Case #4 As we have two variables (x_0 and x_1), we have the possibility to add another criterion to the approximation, such as the volume and the surface values of L_p and the star object have to be the same instead of the

p	x_0	x_{1a}	x_{1b}
0	0	0	0
$\frac{1}{10}$	$\frac{1}{1024}$ 0.00098	$\frac{1}{2047}$ 0.00049	$\frac{1}{59049}$ 1.69351e-05
$\frac{2}{10}$	$\frac{1}{32}$ 0.03125	$\frac{1}{63}$ 0.01587	$\frac{1}{243}$ 0.00412
$\frac{3}{10}$	$\frac{\sqrt[3]{4}}{16}$ 0.09921	$-\frac{\sqrt[3]{4}}{\sqrt[3]{4}-32}$ 0.05220	$\frac{\sqrt[3]{9}}{81}$ 0.02568
$\frac{4}{10}$	$\frac{\sqrt{2}}{8}$ 0.17678	$-\frac{\sqrt{2}}{\sqrt{2}-16}$ 0.09696	$\frac{\sqrt{3}}{27}$ 0.06415
$\frac{5}{10}$	$\frac{1}{4}$ 0.25000	$\frac{1}{7}$ 0.14286	$\frac{1}{9}$ 0.11111
$\frac{6}{10}$	$\frac{\sqrt[3]{2}}{4}$ 0.31498	$-\frac{\sqrt[3]{2}}{\sqrt[3]{2}-8}$ 0.18693	$\frac{\sqrt[3]{3}}{9}$ 0.16025
$\frac{7}{10}$	$\frac{\sqrt[3]{16}}{4}$ 0.37150	$-\frac{\sqrt[3]{16}}{\sqrt[3]{16}-8}$ 0.22812	$\frac{\sqrt[3]{81}}{9}$ 0.20816
$\frac{8}{10}$	$\frac{\sqrt[3]{8}}{4}$ 0.42045	$-\frac{\sqrt[3]{8}}{\sqrt[3]{8}-8}$ 0.26618	$\frac{\sqrt[3]{27}}{9}$ 0.25328
$\frac{9}{10}$	$\frac{\sqrt[3]{256}}{4}$ 0.46294	$-\frac{\sqrt[3]{256}}{\sqrt[3]{256}-8}$ 0.30118	$\frac{\sqrt[3]{3^8}}{9}$ 0.29503
1	1/2 0.50000	1/3 0.33333	1/3 0.33333

Table 1: Approximating L_p (fully covering case).

compactness ratio. We can hold this criterion by having $0 \leq x_{1d} \leq x_{0d} \leq 1/2$:

$$V_{L_p} = V_{obj}, \quad S_{L_p} = S_{obj}, \quad (15)$$

where $0 \leq x_{0d}, x_{0d} \leq 1/2, 0 \leq x_{1d}, x_{1d} \leq x_{0d}$.

5.2.2. Details of case #2

We take the slice of the approximating object created with the plane defined by any two axes at origin, then we take the positive octant from this object. From ¹¹ we know that the approximation is made by a star. The approximating star can be expressed in terms of the slope m' of the line l_h defined by the more horizontal side of the star:

$$l_h(x) = xm' - m'.$$

The object used by the special approximation can be described trivially as a function of the slope m' .

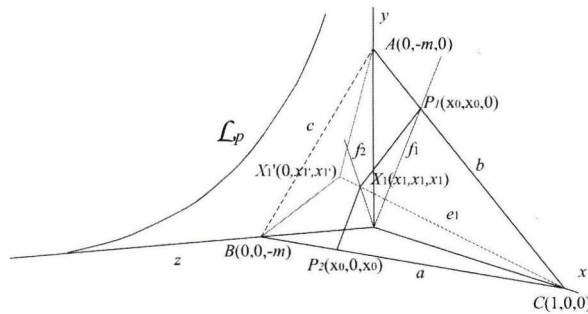


Figure 7: Finding x_{1b} by analysing the positive octant in 3D. C is a node of the approximating object, P_1 and P_2 are the intersecting points of two planes of the object, and X_1 is the intersecting point of all the six planes of the object in the octant. $f_1 : x = y$, and $z = 0$, $f_2 : x = y = z$.

We analyse the 3D approximation in the positive octant, see Figure 7. Point C is one of the nodes of the star-object, while points P_1 and P_2 are the intersections of two of the three planes on the planes defined by axes $\langle x, y \rangle$ and $\langle x, z \rangle$, respectively. Lines a and b is the extension of the segments $\langle C, P_1 \rangle$, $\langle C, P_2 \rangle$. These lines have intersection points A and B with the plane defined by the axis $\langle y, z \rangle$. The derived 2D approximation has the slope $m' = -1$. In that case the star must approximate the 2D L_p circle with $r = -m$. Note that m is the slope-property of the original 3D approximating object.

$$f(x) = m'x - m'r, \quad x = \frac{m'r}{m' - 1}, \quad (16)$$

$$x'_1 = -\frac{m^2}{m - 1}. \quad (17)$$

Our goal is to have an approximation where $m = m'$. Thus, by an $r = -m$ substitution in (16), we have (17), namely the point $X'_1(x'_1, x'_1, 0)$. In order to get the proper X_1 point,

we have to compute the intersection of the lines defined as $\langle X'_1, C \rangle$ and by the $x = y = z$ equality:

$$x_1 = \frac{x'_1}{x'_1 + 1} = \frac{m^2}{m^2 - m + 1}.$$

With the substitution $m = x/(x - 1)$, we have x_1 expressed in terms of x_0 :

$$\frac{x_0^2}{x_0^2 - x_0 + 1}.$$

5.3. Defining weights for chamfering

For defining the weights for the chamfering, we have to calculate both the slope m of the 3D object, and the slope m' of the refinement. For the known x_0 , we can have

$$m = \frac{x_0}{x_0 - 1}.$$

For computing m' , we have to compute X'_1 first. For this, we can leave the third coordinate of X_1 . We introduce m_0 as the slope of e'_1 – the projection of the line to 2D. For m_0 , we have:

$$m_0 = \frac{x_1}{x_1 - 1}.$$

For the coordinates of intersection point of e'_1 and the plane of $\langle y, z \rangle$, we have $(0, -m_0, 0)$. That is, we have $X'_1(0, -m_0, -m_0)$. We have the points $B(0, 0, -m)$ and $X_1(0, -m_0, -m_0)$, thus we can easily compute m' :

$$m' = \frac{(-m)(-m_0)}{-m_0 - 1}.$$

The weights for chamfering are now as follows:

$$\begin{aligned} WX_1 &= 1, & WY_1 &= \frac{1}{-m}, & WZ_1 &= \frac{1}{mm'} \\ WX_2 &= 1, & WY_2 &= \frac{1}{mm'}, & WZ_2 &= \frac{1}{-m} \\ WX_3 &= \frac{1}{-m}, & WY_3 &= 1, & WZ_3 &= \frac{1}{mm'} \\ WX_4 &= \frac{1}{mm'}, & WY_4 &= 1, & WZ_4 &= \frac{1}{-m} \\ WX_5 &= \frac{1}{-m}, & WY_5 &= \frac{1}{mm'}, & WZ_5 &= 1 \\ WX_6 &= \frac{1}{mm'}, & WY_6 &= \frac{1}{-m}, & WZ_6 &= 1 \end{aligned}$$

6. Results

In Table 2, we summarize the results for x_0 and x_1 via several type of approximations.

See Figure 8 for results of overlapping case and for the properties of the approximating object. For computing the volume and the surface of the object, we used the x_0 and x_1 values. Thus, we can compare the properties of L_p and the star-object.

p	Covering case				Overlapping case							
	Special case		General case		Case #1		Case #2		Case #3		Case #4	
	x_{0_s}	x_{1_s}	x_{0_g}	x_{1_g}	x_{0_a}	x_{1_a}	x_{0_b}	x_{1_b}	x_{0_c}	x_{1_c}	x_{0_d}	x_{1_d}
0.0	0	0	0	0	0	0	0	0	0	0	0	0
0.1	0.00098	0.00049	0.00098	0.00017	2.57e-07	1.285e-07	0.00009	8.155e-09	2.239e-08	1.693e-05	5.437e-06	3.313e-08
0.2	0.03125	0.01587	0.03125	0.00412	0.00109	0.00055	0.01346	0.00018	0.00039	0.00412	0.00410	0.00032
0.3	0.09921	0.05220	0.09921	0.02568	0.01662	0.00838	0.06711	0.00480	0.00878	0.02568	0.03447	0.00635
0.4	0.17677	0.09696	0.17678	0.06415	0.06248	0.03225	0.14395	0.02363	0.03916	0.06415	0.09514	0.02749
0.5	0.25	0.14286	0.25	0.11111	0.13377	0.07168	0.22233	0.05976	0.09313	0.11111	0.17021	0.06528
0.6	0.31498	0.18693	0.31498	0.16025	0.21620	0.12120	0.29346	0.10864	0.16337	0.16025	0.24718	0.11483
0.7	0.37150	0.22812	0.37150	0.20816	0.29834	0.17532	0.35573	0.16417	0.24241	0.20816	0.32008	0.17019
0.8	0.42045	0.26618	0.42045	0.25328	0.37414	0.23012	0.41002	0.22176	0.32572	0.25328	0.38672	0.22675
0.9	0.46294	0.30118	0.46294	0.29503	0.44139	0.28320	0.45769	0.27864	0.41153	0.29503	0.44667	0.28162
1.0	0.5	0.33333	0.5	0.33333	0.5	0.33333	0.5	0.33333	0.5	0.33333	0.5	0.33333

Table 2: Results of approximations for several p values.

7. Conclusion, applications

Since we have chamfering for computing distance in digital space, we can use this rapid approximation in other fields, too. In database image retrieval there are usually more features, which can be used for comparing images. These features can be e.g. color, shape and texture. With our approximating approach, we can find images which are close to each other in their color feature, but they can be far in texture or shape.

Acknowledgement

This research was partly supported by the János Bolyai grant of the Hungarian Academy of Sciences.

References

- G. Borgefors, Distance transformations in arbitrary dimensions, *Comput. Vision Graphics Image Process.* 27 (1984) 321–345. 1, 2, 3
- P. Danielsson, 3D octagonal metrics, *Eighth Scandinavian Conf. Image Process.* (1993) 727–736. 2
- P. Das, P. P. Chakrabarti, B. N. Chatterji, Distance functions in digital geometry, *Inform. Sci.* 42 (1987) 113–136. 2
- P. Das, P. P. Chakrabarti, B. N. Chatterji, Generalised distances in digital geometry, *Inform. Sci.* 42 (1987) 51–67. 2
- P. Das, B. N. Chatterji, Octagonal distances for digital pictures, *Inform. Sci.* 50 (1990) 123–150. 2
- A. Fazekas, Lattice of distances based on 3d-neighbourhood sequences, *Acta Math. Acad. Paedagog. Nyházi.* 15 (1999) 55–60. 2
- A. Fazekas, A. Hajdu, L. Hajdu, Lattice of generalised neighbourhood sequences in nD and ∞D, *Publicationes Mathematicae* 60 (2002) 405–427. 2
- A. Hajdu, Geometry of neighbourhood sequences, *Pattern Recognition Lett.* 24 (15) (2003) 2597–2606. 2

- A. Hajdu, L. Hajdu, Approximating the euclidean distance using non-periodic neighbourhood sequences, *Discrete Math.* 283 (1-3) (2004) 101–111. 2
- A. Hajdu, L. Hajdu, R. Tijdeman, General neighborhood sequences in \mathbb{Z}^n , *Discrete Appl. Math.* 155 (2007), 2507–2522. 2
- A. Hajdu, T. Tóth: Approximating non-metrical Minkowski distances in 2D, *Pattern Recognition Lett.* 29/6 (April 2008), 813-821. 1, 4, 6
- B. Nagy, Distance with generalised neighbourhood sequences in nD and ∞D, *Discrete Appl. Math.* 156 (2008), 2344–2351. 2
- B. Nagy, Distances with neighbourhood sequences in cubic and triangular grids, *Pattern Recognition Letters* 28 (2007) 99–109. 2
- B. Nagy, R. Strand, Distances based on neighbourhood sequences in non-standard three-dimensional grids, *Discrete Appl. Math.* 155 (4) (2007) 548–557. 2
- A. Rosenfeld, J. Pfaltz, Distance functions on digital pictures, *Pattern Recognition* 1 (1968) 33–61. 1, 2
- T. Schouten, E. van den Broek, Fast exact euclidean distance (FEED) transformation, *ICPR* 3 (2004) 594–597. 3
- F. Y. Shih, Y.-T. Wu, Fast euclidean distance transformation in two scans using a 3X3 neighborhood, *CVIU* 93 (2) (February 2004) 195–205. 3
- M. Yamashita, T. Ibaraki, Distances defined by neighbourhood sequences, *Pattern Recognition* 19 (1986) 237–246. 2

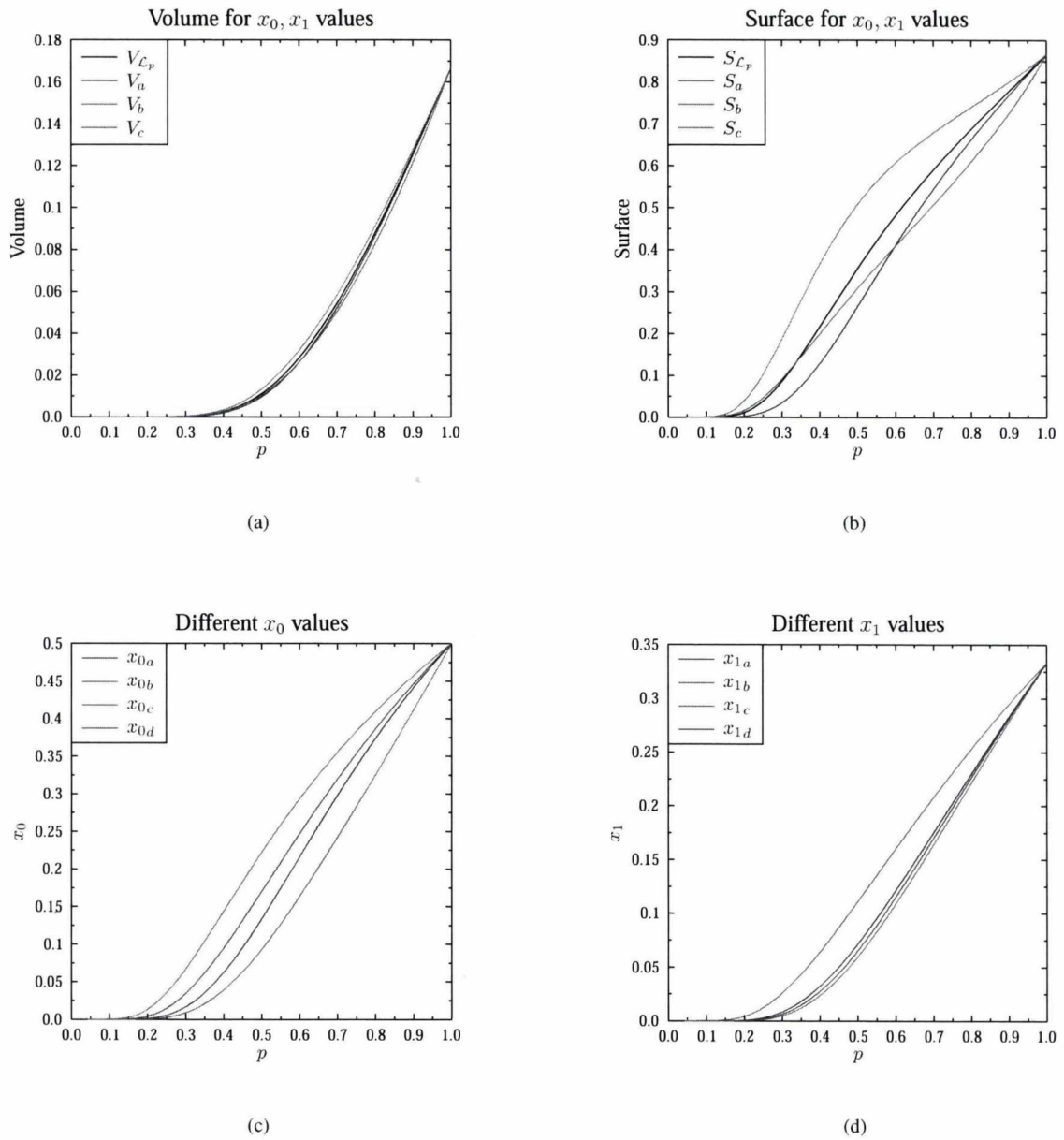


Figure 8: The results of the approximation in the overlapping case. The surface and volume values are computed with the proper x_0 and x_1 values computed in the corresponding approximation.

L_2 -Optimal Weak-Perspective Camera Calibration

Levente Hajder¹

¹ Geometric Modelling and Computer Vision Laboratory
Computer and Automation Research Institute
Budapest, Hungary

Abstract

Camera calibration is a key problem in 3D/4D computer vision. Several efficient algorithms have been published since the late 80's which try to minimize the error of the calibration with respect to the camera parameters. Most of them deal with the so-called (perspective) pinhole camera models. This is not a simple goal: the problem is nonlinear due to the perspectivity. The strategy of these methods is that they estimate the camera parameter imprecisely first, then accurate parameters are obtained by numerical optimization. Therefore, these methods are relatively slow. In this paper, we show that the weak-perspective camera model can be optimally calibrated if the L_2 norm is used. The solution is given by a closed-form formula, thus the estimation is very fast. We show that the proposed calibration method can be utilized in the problem of 3D reconstruction with missing data. It is also discussed that the scaled orthographic camera parameters can be estimated in an iterative way.

Categories and Subject Descriptors (according to ACM CCS): I.4.8 [Image Processing and Computer Vision]: Scene UnderstandingMotion

1. Introduction

Camera calibration is a key problem in 3D/4D computer vision. Several efficient algorithms have been published since the late 80's. There are well-known solutions^{5,16} to calibrate the perspective camera. These methods give a rough estimation of the parameters first, then this estimation is refined using numerical optimization. The affine, weak-perspective and scaled orthographic camera model have not been considered separately, to our best knowledge. They are only considered as auto-calibration problems. Mathematically, the problem is a factorization one: the so-called measurement matrix should be factorized into the product of the camera and the structure parameters.

The classical factorization method for the full case – when the measurement matrix is factorized into 3D motion and structure matrices – was developed by Tomasi and Kanade¹⁴ in 1992. The weak-perspective extension was published by Weinshall and Kanade¹⁵. The factorization was extended to the paraperspective¹⁰ case as well as to the real perspective¹³ one.

The problem of missing data has already been addressed by Tomasi and Kanade¹⁴. They proposed a naive approach

which transforms the missing data problem to the full matrix factorization by estimating the missing entries. Shum et al.¹² proposed a method to reconstruct the objects from range images. Their method is successfully applied to the Structure from Motion (SfM) problem by Buchanan et al.³.

The mainstream idea to the factorization with missing data is to factorize the rank 4 measurement matrix into affine structure and motion matrices which are of size 4. (The Shum-method^{12,3} also computes affine structure and motion matrices, but the size of those matrices is 3.) This can be done by the mathematical method called Principal Component Analysis with Missing Data (PCAMD). This problem has been already pointed out by mathematicians since the middle 70's¹¹. These methods can be applied directly to the SfM problem as it is written in³. Hartley & Schafalitzky⁶ proposed the PowerFactorization method which is based on the Power method. Power method is an iteration to compute the dominant n -dimensional subspace of a given matrix. Buchanan & Fitzgibbon⁴ handled the problem as an alternation consisting of two nonlinear iterations to be solved. They suggested using the Damped-Newton method with line search to compute the optimal structure and motion matrices. We have also proposed a factorization method⁹

which assumes scaled orthographic camera. This algorithm is repeated here in brief. To our best knowledge, there is no other factorization method which deals with scaled orthographic or weak perspective camera models.

We consider the weak-perspective camera calibration and factorization in this paper. The weak-perspective methods can be used for perspective reconstruction if the elements of the measurement matrix are multiplied with the corresponding projective depths as it is proposed by Sturm et al.¹³, or the perspective camera parameters can be estimated from weak-perspective camera reconstruction. Finally, the results can be refined by the well-known bundle adjustment method².

2. Problem statement

Given the 3D coordinates of a static object's points and the 2D coordinates of their projection in the image, the aim of the camera calibration is to estimate the camera parameters which represent the 3D \rightarrow 2D projection.

Let us denote the 3D coordinates of the i^{th} point by X_i , Y_i , and Z_i . The corresponding 2D coordinates are denoted by u_i , and v_i . The perspective (pinhole) camera projection is usually written as follows

$$\begin{bmatrix} u_i \\ v_i \\ 1 \end{bmatrix} \sim C[R|T] \begin{bmatrix} X_i \\ Y_i \\ Z_i \\ 1 \end{bmatrix}. \quad (1)$$

where R is the rotation (orthonormal) matrix, and T is the translation vector between the world's and camera's coordinate systems. (These parameters are usually called the extrinsic parameters of the perspective camera.) The ' \sim ' sign denotes equality up to scale. The intrinsic parameters of the camera are stacked in the upper triangular matrix C ⁵.

If the depth of object is much smaller than the distance between the camera and the object, the weak-perspective camera model is a good approximation:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = [M|t] \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix}. \quad (2)$$

where M is the motion matrix consisting of two vectors as $M = [m_1, m_2]^T$, t is a 2D offset vector which locates the position of the world's origin in the image. In the weak-perspective camera model, the rows of the motion matrix are not allowed to be arbitrary, they must satisfy the orthogonality constraint $m_1^T m_2 = 0$. A special case of the weak-perspective camera model is the scaled orthographic one, when $m_1^T m_1 = m_2^T m_2$.

If the affine camera is considered, there is no constraint: the elements of the motion matrix M may be arbitrary.

3. Camera Calibration

In this section, the weak-perspective, the scaled orthographic, and the affine camera calibration is discussed. The goal of the calibration is to minimize the reprojection error in the least squares sense. The reprojection error is written as

$$\sum_{i=1}^N \left\| \begin{bmatrix} u_i \\ v_i \end{bmatrix} - [M|t] \begin{bmatrix} X_i \\ Y_i \\ Z_i \end{bmatrix} \right\|^2 \quad (3)$$

where N is the number of points to be considered in the calibration, $\|\cdot\|$ denotes the L_2 (Euclidean) vector norm. As Horn et al.^{7,8} proved, the translation vector t is optimally estimated if it is selected as the center of gravity of the 2D points. These are easily calculated as $\bar{u} = 1/N \sum_{i=1}^N u_i$ and $\bar{v} = 1/N \sum_{i=1}^N v_i$.

3.1. Weak-perspective projection

If the weak-perspective camera model is assumed, the error define in eq. 3 can be rewritten in a more compact form as

$$\|w_1 - m_1^T S\|^2 + \|w_2 - m_2^T S\|^2 \quad (4)$$

where

$$w_1 = [u_1 - \bar{u}, u_2 - \bar{u}, \dots, u_N - \bar{u}], \quad (5)$$

$$w_2 = [v_1 - \bar{v}, v_2 - \bar{v}, \dots, v_N - \bar{v}], \quad (6)$$

$$S = \begin{bmatrix} X_1 & X_2 & \dots & X_P \\ Y_1 & Y_2 & \dots & Y_P \\ Z_1 & Z_2 & \dots & Z_P \end{bmatrix}. \quad (7)$$

If the Lagrange multiplier λ is introduced, the weak-perspective constraint can be considered. The error function is modified as follows

$$\|w_1 - m_1^T S\|^2 + \|w_2 - m_2^T S\|^2 + \lambda m_1^T m_2 \quad (8)$$

The optimal solution of this error function is given by its derivatives with respect to λ , m_1 , and m_2 :

$$m_1^T m_2 = 0 \quad (9)$$

$$SS^T m_1 - Sw_1 + \lambda m_2 = 0 \quad (10)$$

$$SS^T m_2 - Sw_2 + \lambda m_1 = 0 \quad (11)$$

m_2 is easily expressed from eq. 10 as

$$m_2 = \frac{1}{\lambda} (Sw_1 - SS^T m_1). \quad (12)$$

If one substitutes m_2 into eq. 11, and 9, then the following expressions are obtained:

$$\frac{1}{\lambda} SS^T (Sw_1 - SS^T m_1) - Sw_2 + \lambda m_1 = 0 \quad (13)$$

$$\frac{1}{\lambda} m_1^T (Sw_1 - SS^T m_1) = 0 \quad (14)$$

If eq. 13 is multiplied by λ , then m_1 can be expressed as

$$m_1 = \left(SS^T SS^T - \lambda^2 I \right)^{-1} \left(SS^T S w_1 - \lambda S w_2 \right) \quad (15)$$

where I is the 3×3 identity matrix. If the expressed m_1 is substituted into eq. 14, the equation from which λ should be determined is obtained:

$$\frac{1}{\lambda} A^T(\lambda) B^{-T}(\lambda) \left(S w_1 - SS^T B^{-1}(\lambda) A(\lambda) \right) = 0 \quad (16)$$

where

$$A(\lambda) = SS^T S w_1 - \lambda S w_2 \quad (17)$$

$$B(\lambda) = SS^T SS^T - \lambda^2 I \quad (18)$$

$$(19)$$

$A(\lambda)$ and $B(\lambda)$ are matrices which have elements containing polynomials of unknown variable λ . This kind of matrices are called matrix of polynomials in this study. The difficulty is that matrices $A(\lambda)$ and $B(\lambda)$ should be inverted. However, this inversion can be written as a fraction of two matrices. For example, $B^{-1}(\lambda)$ can be written as

$$B^{-1}(\lambda) = \frac{\text{adj} \left(SS^T SS^T - \lambda^2 I \right)}{\det \left(SS^T SS^T - \lambda^2 I \right)} \quad (20)$$

where $\text{adj}(\cdot)$ denotes the adjoint[†] of a matrix. It is trivial that $\det(B(\lambda))$ is a polynomial of λ , while $\text{adj}(B(\lambda))$ is a matrix of polynomials. This expression is useful since the equation can be multiplied by the determinants of $B(\lambda)$ and $A(\lambda)$.

If one makes elementary modifications, eq. 16 can be rewritten as

$$\frac{A^T(\lambda) \text{adj} B^T(\lambda)}{\det B(\lambda)} \frac{\det B(\lambda) S w_1 - SS^T \text{adj} B(\lambda) A(\lambda)}{\det B(\lambda)} = 0 \quad (21)$$

It is trivial as well that eq. 21 is true if the nominator equals to zero. The Lagrangian parameter λ is calculated by solving the polynomial as follows

$$A^T(\lambda) \text{adj} B^T(\lambda) \left(\det B(\lambda) S w_1 - SS^T \text{adj} B(\lambda) A(\lambda) \right) = 0. \quad (22)$$

This final equation is a polynomial of degree 10. It has 10 complex solutions, but only the real values must be considered. The obtained real values of λ should be substituted into eq. 15 and the yielded m_1 and λ into eq. (12). Then the optimal solution is the one which minimizes the reprojection error in eq. 3.

[†] The transpose of the adjoint is also called as the matrix of cofactors.

3.2. Scaled Orthography

Scaled orthography is a special type of weak perspective projection. While the weak-perspective model assumes that the two base vectors m_1 and m_2 are orthogonal, a novel assumption is introduced in the scaled orthographic projection model: the length of the base vectors are the same ($m_1^T m_1 = m_2^T m_2$).

Unfortunately, we have not found a closed-form solution which minimizes the reprojection error. We have proposed the following trick in our paper⁹ published in 2008. If an initial value of the motion matrix is given, the base vectors of the motion matrix M can be completed with a third row m_3 : its row is orthogonal to the first two rows, its length is the average of those. Then the motion matrix is a scaled orthonormal one: $M = qR$. The third projected coordinates can be estimated as $w_3 = m_3^T S$. Let us introduce the measurement matrix $W = [w_1^T, w_2^T, w_3^T]^T$, then the reprojection error is given as follows

$$\|W - qRS\|_F^2. \quad (23)$$

where $\|\cdot\|_F$ denotes the Frobenius-norm of a matrix[‡]. This is the well-known registration problem of 3D pointsets in matrices S and M . The optimal solution has been already published^{1, 7, 8} except the estimation of the scale⁹ which is given in the appendix. (The appendix describes the computation of the optimal translation and rotation as well.)

3.3. Affine Camera

The affine camera is the simplest one: the projection is represented by a 4×4 matrix with no constraints. This matrix represents the camera orientation as well as the translation and affine distortion. The reprojection error is given as follows

$$\|W - M_{\text{aff}} S_{\text{aff}}\|_F^2, \quad (24)$$

where M_{aff} is the affine motion matrix, S_{aff} is the $4 \times P$ affine structure matrix. The calibration problem is linear with respect to the elements of M_{aff} . The optimal solution is calculated as follows.

$$M_{\text{aff}} = WS^\dagger \quad (25)$$

where \dagger denotes the so-called Moore-Penrose pseudoinverse.

4. Structure from Motion with Missing Data

In this section, it is described how can the above discussed methods apply to the factorization problem.

[‡] The Frobenius norm of a matrix is the sum of the square of all elements of the matrix.

The proposed method is a down-hill alternation to minimize the reprojection error defined as follows

$$\left\| H \odot \left(W - [M|t] \begin{bmatrix} S \\ 1 \end{bmatrix} \right) \right\|_F^2, \quad (26)$$

where M is the motion matrix consisting of the camera parameters in every frame, structure matrix S contains the 3D coordinates of the points. (Points are located in the columns of matrix S .) ' \odot ' denotes the so-called Hadamard product[§], and H is a the mask matrix. If H_{ij} is zero, then the j^{th} point in the i^{th} frame is not visible (missing). If $H_{ij} = 1$, the point is visible. Remark that the motion matrix M in this formula consists of the motion matrices of all the frames.

Algorithm 1 Summary of scaled orthographic factorization

$M^{(0)}, t^{(0)}, S^{(0)} \leftarrow$ Parameter Initialization
 $\tilde{H}, \tilde{W}^{(0)}, \tilde{M}^{(0)}, \tilde{t}^{(0)} \leftarrow$ Complete($H, W, M^{(0)}, t^{(0)}, S^{(0)}$)
 $k \leftarrow 0$
repeat
 $k \leftarrow k + 1$
 $\tilde{M}^{(k)} \leftarrow$ M-Step($\tilde{H}, \tilde{W}^{(k)}, S^{(k-1)}$)
 $\tilde{W}^{(k)} \leftarrow$ Complete($W, \tilde{H}, \tilde{M}^{(k)}, S^{(k-1)}$)
 $S^{(k)} \leftarrow$ S-Step($\tilde{H}, \tilde{W}^{(k)}, \tilde{M}^{(k)}$)
 $\tilde{W}^{(k)} \leftarrow$ Complete($W, \tilde{H}, \tilde{M}^{(k)}, S^{(k)}$)
until $\left\| \tilde{H} \odot \left(\tilde{W}^{(k)} - [\tilde{M}^{(k)}|t^{(k)}] \begin{bmatrix} S^{(k)} \\ 1 \end{bmatrix} \right) \right\|_F^2$ con-
 verges.

Each cycle of the proposed methods is divided into the following main steps:

1. M-step. The aim of this step is to compute the motion matrix $M = [M_1^T, M_2^T, \dots, M_1^T]^T$ and translation vector $t = [t_1^T, t_2^T, \dots, t_1^T]^T$, where the index denotes the frame number. It is trivial that the estimation of these submatrices are independent from each other if the elements of the structure matrix S are fixed. The optimal solution is given by one of the camera calibration algorithms given in section 3. Note that missing data should be skipped in the estimation.
2. S-step. The goal of the S-step is to compute the structure matrices if the elements of the motion matrix and the translation error are fixed. The 3D points represented by the columns of structure matrix must be computed independently. (They are independent from each other.) Missing data should not be considered in the estimation of course. The optimal solution is given by the Moore-Penrose pseudoinverse.

The summary of the proposed algorithm is given in Alg. 1 and 2. Alg. 1 shows the skeleton of the factorization for the scaled orthographic camera, while alg. 2 that of affine

[§] $A \odot B = C$ if $c_{ij} = a_{ij} \cdot b_{ij}$.

and weak-perspective ones. The difference is that the motion matrices and the measurement matrix should be completed if the scaled orthographic camera is assumed as it is described in sec. 3.

Algorithm 2 Summary of affine / weak-perspective (SO) factorization

$M^{(0)}, t^{(0)}, S^{(0)} \leftarrow$ Parameter Initialization
 $k \leftarrow 0$
repeat
 $k \leftarrow k + 1$
 $M^{(k)} \leftarrow$ M-Step($H, W, S^{(k-1)}$)
 $S^{(k)} \leftarrow$ S-Step($H, W, M^{(k)}$)
until $\left\| H \odot \left(W - [M^{(k)}|t^{(k)}] \begin{bmatrix} S^{(k)} \\ 1 \end{bmatrix} \right) \right\|_F^2$ converges.

5. Parameter initialization

The proposed factorization method requires initial values of the matrices. The key idea of our initialization is that the factorization with missing data can be divided into full matrix factorizations of submatrices as we proposed in ⁹.

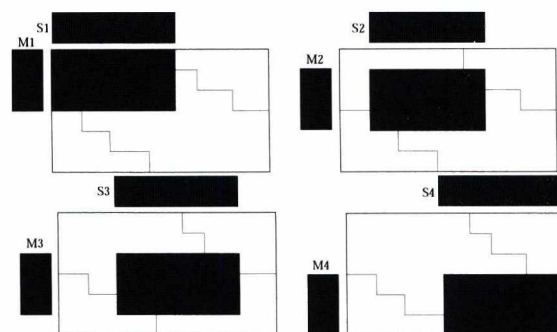


Figure 1: Problem divided into factorization of submatrices.

The Tomasi-Kanade factorization needs at least 3 frames to compute structure and motion. Therefore, the selected feature points should be visible in frames 1, 2, and 3 as it is visualized in the left image of Fig. 1. Then the Tomasi-Kanade factorization¹⁴ with the extension of Weinshall and Kanade¹⁵ is run with the selected feature points. Motion and structure matrices (M_1 and S_1) and offset vector t_1 are obtained by full matrix factorization. Then another full matrix is formed with the feature points visible in frames 2, 3, and 4. By applying the factorization, motion matrix M_2 , structure matrix S_2 , and offset vector t_2 are computed. Matrices M_3 , S_3 , and offset vector t_3 are computed from the points visible in frames 3, 4, and 5. This process is repeated until the matrices M_{F-2} , S_{F-2} , and t_{F-2} are obtained. The steps of parameter initialization are summarized in Algorithm 3. The key problem of the initialization is to register the results of the new factorization to the already registered ones which

Algorithm 3 Parameter Initialization

```

 $W_1 \leftarrow$  Feature points common in frames 1,2,3.
 $M_1, S_1, t_1 \leftarrow$  TomasiKanade( $W_1$ )
for  $i = 2$  to  $(F - 2)$  do
   $W_i \leftarrow$  Common feature points of frames  $i, i + 1, i + 2$ .
   $M_i, S_i, t_i \leftarrow$  TomasiKanade( $W_i$ )
  Put points common in  $S$  and  $S_i$  into  $S'$  and  $S'_i$ , respectively.
  Register matrices  $S'$  and  $S'_i$  to each other.
  Update matrices  $M$  and  $S$  and offset vector  $t$ 
end for

```

are contained by matrices M , S , and vector t . New factorization in the i^{th} cycle can be written as

$$[W_i | t_i] \begin{bmatrix} S_i \\ 1 \end{bmatrix}. \quad (27)$$

The feature points common in S and S_i are denoted by S' , and S'_i , respectively. These point sets should be registered to each other by the method described in the appendix. If s'^{ij} and $s_i'^{ij}$ denote the j^{th} elements of the point sets, the registration is given by

$$s'^{ij} \approx qR(s_i'^{ij} - o_2) + o_1, \quad (28)$$

where q , R , o_1 , and o_2 are the scale factor, rotation matrix, center of gravity of the first, and that of the second point set, respectively. The original structure matrix S_i should be transformed as well. After transformation, the new points are added to the point set S .

The last two rows of M_i and t_i are inserted at the end of M , and t , respectively, if the matrices M_i and t_i are transformed as follows:

$$M_i \leftarrow \frac{1}{q} M_i R^T, \quad (29)$$

$$t_i \leftarrow t_i + M_i o_2 - \frac{1}{q} M_i R^T o_1. \quad (30)$$

The parameter initialization can be applied for both the weak-perspective and the scaled orthographic camera models. The difference is only that different motion constraints are considered in the factorization as it is proposed by Weinshall et al. ¹⁵.

The affine parameter initialization is very similar. There are two differences:

- The factorization of the submatrices are carried out by a Singular Value Decomposition. The four most dominant singular value, and the corresponding vectors should be kept.
- The registration of the factorized submatrices is a linear problem. It is done using the pseudoinverses of the common matrices.

6. Tests on synthesized data

Several experiments with synthetic data have been carried out to study the properties of the proposed methods. Three methods have been compared: (i) **SO** Scaled Orthographic factorization, (ii) **WP** Weak-Perspective factorization, (iii) **AFF**: Affine factorization.

We have examined three properties of the reconstruction as follows.

1. Reconstruction error: The reconstructed 3D points are registered to the generated (ground truth) ones as it is described in the appendix. This registration error is called reconstruction error in the tests.
2. Motion error: The row vectors of the obtained 3D motion matrix can be registered to that of the generated (ground truth) motion matrix. This registration error gives the error of the motion.
3. Time demand: The running time of each algorithm is measured. The given values contain every step from the parameter initialization to the final reconstruction.

In order to compare the affine methods listed above to the proposed alternation algorithm, the computation of the metric 3D structure is carried out by the classical weak-perspective Tomasi-Kanade factorization. The $2F \times 4$ affine motion is multiplied by the $4 \times P$ affine structure matrix, and a full measurement matrix is obtained. Then this measurement matrix is factorized by Tomasi-Kanade algorithm ¹⁴ with the Weinshall-Kanade ¹⁵ extension.

All of the methods were implemented in Java. (The code will be available from the Internet soon.) The tests were run on an Intel Core4Quad 2.33 GHz PC with 4 GByte memory. The implementation is a single threaded one, we plan to make it multi threaded in the future.

6.1. Generation of the measurement matrix

The input (measurement matrix) is composed of 2D trajectories. These trajectories are generated in the following way: (i) Random three-dimensional coordinates are generated by a zero-mean Gaussian random number generator with variance σ_{3D} . (ii) The generated 3D points are rotated by random angles. (iii) Points are projected using scaled orthographic projection. (The scale parameter itself are generated by a continuous uniform distribution with boundaries 0.5, and 1.5.) (iv) Noise is added to the projected coordinates. It is generated by a zero-mean Gaussian random number generator as well. Its variance is set to σ_{2D} . (v) Finally, the measurement matrix W is composed of the projected points. (vi) Motion and structure parameters are initialized as it is described in Sec. 5. For each test case, 25 measurement matrices are generated and the rival reconstruction methods are executed 25 times. The results shown in this section is calculated as the average of these 25 executions.

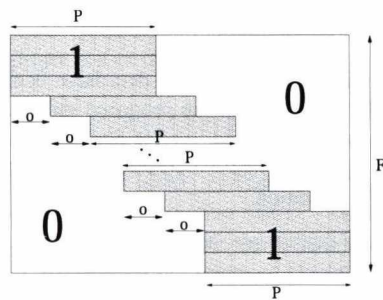


Figure 2: Structure of mask matrix.

6.2. Generation of the mask matrix

The mask generator algorithm has three parameters: (i) P : Number of the visible points in each frame, (ii) F : Number of frames, (iii) O : Offset between the neighboring frames. For our synthetic tests, offset was set to $O = 2$.

The structure of the generated mask is illustrated in Fig. 2. The missing data ratio in the matrix is calculated by the following equation: $md_{\%} = 100PF / (F(P + (F - 5)O))$.

General remarks. The tests show that the SO algorithm outperforms the other methods in every test case. This is not surprising since the test data have been generated using scaled orthographic projection. This is true for the reconstruction error as well as the motion error. The second place in accuracy is given to the proposed weak-perspective (WP) method which is always better than the affine one, but it is slightly less accurate comparing to the SO method.

Examining the time demand, it is not clear which is the fastest and the slowest method. The WP algorithm is always faster than the SO one, but the affine (AFF) method can be faster as well as slower than the rival algorithms. The affine factorization is slow when there are huge amount of input data. It is because a full factorization¹⁴ must be applied after the affine factorization to obtain metric reconstruction, and this can be very slow due to the Singular Value Decomposition. However, this SVD-step can make faster if only the three most dominant singular values and vectors are computed. Unfortunately, the Jama Matrix Package (JAMA) which we used in our implementation does not contain this feature. As it is shown in⁴, there are several methods which implement affine reconstruction. We showed earlier⁹ that the fastest method of those is the so-called Damped-Newton algorithm, which is significantly faster than our affine implementation.

Error versus noise (Figures 3 and 4) The methods are run with gradually increasing noise level. The reconstruction error increases approximately in a linear way for all the methods. The test sequence consists of 100 frames, and $P = 500$ was set. The missing data ratio is 35.72%. The noise level is calculated as $100\sigma_{2D}/\sigma_{3D}$.

The test indicates that the SO algorithm outperforms the ri-

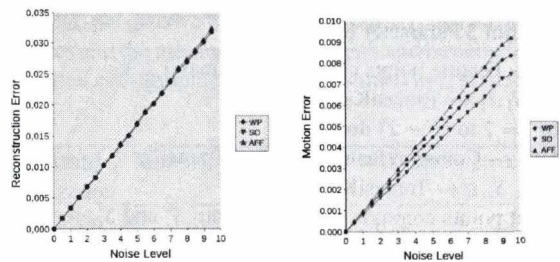


Figure 3: Reconstruction and motion errors and time demand w.r.t. 2D noise.

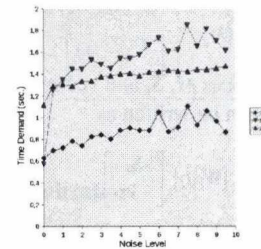


Figure 4: Time demand w.r.t. number of points.

val ones, and the WP method is better than the affine one as it is expected. However, SO needs the most time to finish its execution, and the faster method is the proposed weak-perspective one.

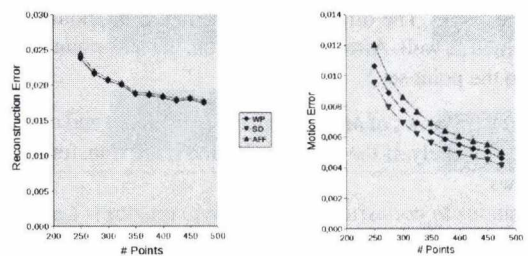


Figure 5: Reconstruction and motion errors and time demand w.r.t. number of points.

Error versus number of points (Figures 5 and 6) P increases from 200 to 500. (The missing data rate decreases from approx. 70% to 40%.) The noise level is 5%, and the sequence consists of 100 frames. The conclusion is similar to the previous test case: the most precise model is given by the SO algorithm, the second one is the WP method. The difference is not significant. However, the weak-perspective reconstruction is much faster than the scaled orthographic one.

Error versus number of frames (Figures 7 and 8) F increases from 20 to 190. The missing data ratio increases from 5% to 70%. The noise level was 5%, and $P = 500$ was

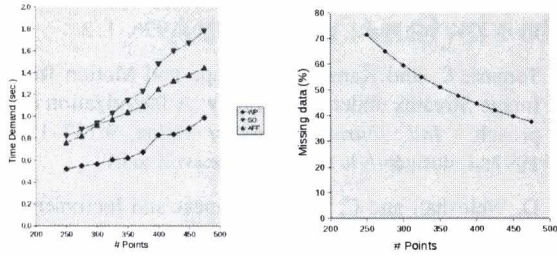


Figure 6: Time demand and missing data ratio w.r.t. number of points.

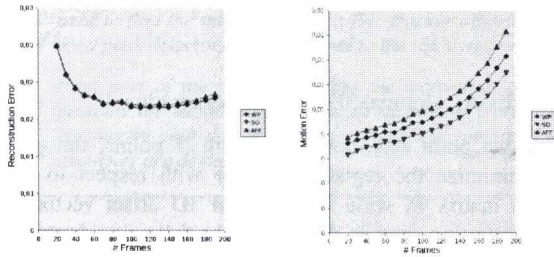


Figure 7: Reconstruction and motion errors and time demand w.r.t. number of frames.

set. In each test case, the most precise algorithm was the one consisting of the scaled orthographic camera model, but it is the slowest one as it is expected. The accuracy of the weak-perspective factorization is better than the affine one both in structure and motion reconstruction.

7. Test on real data

We have tested the proposed algorithms on our 'Cat' sequence. The cat statuette was rotated on a table and 92 photos were taken by a commercial digital camera. The regions of the statuette in the images were automatically determined as it is demonstrated in fig 9.

Feature points were detected using the widely used KLT algorithm, and the points were tracked by correlation-based

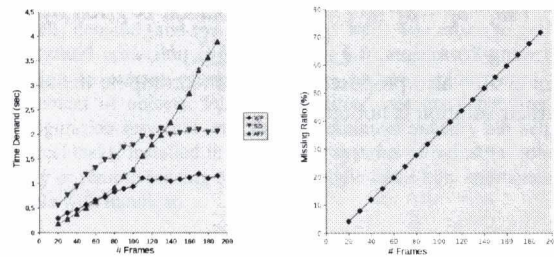


Figure 8: Time demand and missing data ration w.r.t. number of frames.

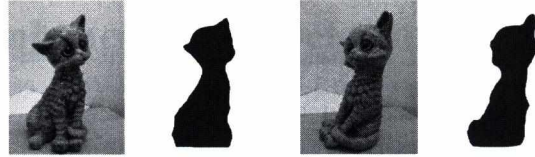


Figure 9: Original images and corresponding masks of 'Cat' sequence.

template matching method. The measurement matrix of the sequence consists of 2290 points and 92 frames. The missing data ratio is 82%.

The 3D reconstructed points are visualized on Fig. 10. We have tested every method and we have compared the time demand of the methods: the running times of the affine, scaled orthographic, and weak-perspective factorization were 484, 199, and 99 seconds, respectively.

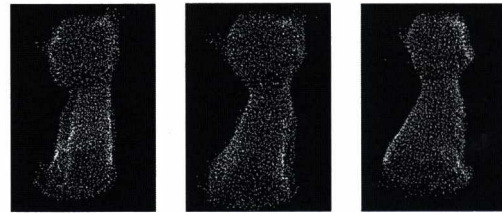


Figure 10: Reconstructed 3D points of 'Cat' sequence. Left: Affine. Center: Scaled Orthographic. Right: Weak-Perspective.

8. Conclusion

We have presented the calibration algorithms of the weak-perspective and scaled orthographic cameras. The calibration of the weak-perspective camera parameters are given by a closed-form formula, while that of the scaled orthographic one is being obtained by an iteration. The proposed methods are successfully applied to the structure from motion with missing data problem. The novel algorithms are more accurate than the state of the art affine reconstruction. The scaled orthographic is a bit more precise and significantly slower than the weak-perspective one.

Acknowledgment. The author would like to express his gratitude to Csaba Kazó and Prof. Dmitry Chetverikov for their council and for providing their mask generator and feature tracking algorithms.

This research has been supported by the NKTH-OTKA grant CK 78409.

References

1. K. S. Arun, T. S. Huang, and S. D. Blostein. Least-squares fitting of two 3-D point sets. *IEEE Trans. on PAMI*, 9(5):698–700, 1987. 3, 8
2. Bill Triggs and Philip McLauchlan and Richard Hartley and Andrew Fitzgibbon. Bundle Adjustment – A Modern Synthesis. In W. Triggs, A. Zisserman, and R. Szeliski, editors, *Vision Algorithms: Theory and Practice*, LNCS, pages 298–375. Springer Verlag, 2000. 2
3. A. M. Buchanan. Investigation into matrix factorization when elements are unknown. Technical report, University of Oxford, <http://www.robots.ox.ac.uk/~amb>, 2004. 1
4. A. M. Buchanan and A. W. Fitzgibbon. Damped newton algorithms for matrix factorization with missing data. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 316–322, 2005. 1, 6
5. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. 1, 2
6. Richard Hartley and Frederik Schaffalitzky. Powerfactorization: 3d reconstruction with missing or uncertain data, 2003. 1
7. B.K.P. Horn. Closed-form Solution of Absolute Orientation using Unit Quaternions. *Journal of the Optical Society of America*, 4:629–642, 1987. 2, 3
8. B.K.P. Horn, H.M. Hilden, and S. Negahdaripour. Closed-form Solution of Absolute Orientation Using Orthonormal Matrices. *Journal of the Optical Society of America*, 5(7):1127–1135, 1988. 2, 3, 8
9. Á. Pernek, L. Hajder, and Cs. Kazó. Metric Reconstruction with Missing Data under Weak-Perspective. In *British Machine Vision Conference*, pages 109–116, 2008. 1, 3, 4, 6
10. C. J. Poelman and T. Kanade. A Paraperspective Factorization Method for Shape and Motion Recovery. *IEEE Trans. on PAMI*, 19(3):312–322, 1997. 1
11. A. Ruhe. Numerical computation of principal components when several observations are missing. Technical report, Departure of Information Processing, Umea Univesity, Sweden, 1974. 1
12. Heung-Yeung Shum, Katsushi Ikeuchi, and Raj Reddy. Principal component analysis with missing data and its application to polyhedral object modeling. *IEEE Trans. Pattern Anal. Mach. Intell.*, 17(9):854–867, 1995. 1
13. P. Sturm and B. Triggs. A Factorization Based Algorithm for Multi-Image Projective Structure and Motion. In *ECCV*, volume 2, pages 709–720, 1996. 1, 2
14. Tomasi, C. and Kanade, T. Shape and Motion from Image Streams under orthography: A factorization approach. *Intl. Journal Computer Vision*, 9:137–154, 1992. 1, 4, 5, 6
15. D. Weinshall and C. Tomasi. Linear and Incremental Acquisition of Invariant Shape Models From Image Sequences. *IEEE Trans. on PAMI*, 17(5):512–517, 1995. 1, 4, 5
16. Z. Zhang. A flexible new technique for camera calibration. *IEEE Trans. on PAMI*, 22(11):1330–1334, 2000. 1

Appendix A: Optimal 3D point set registration

Given two point sets, both containing N points, the goal is to minimize the registration error with respect to the rotation matrix R , scale factor s and 3D offset vector o . The registration error is defined by the following formula: $\sum_{i=1}^N (a_i - (sRb_i + o))^T (a_i - (sRb_i + o))$, where a_i and b_i are the i^{th} 3D vector of the first and second point set, respectively. Horn et al. ⁸ proved that the registration error defined above is minimized optimally with respect to the offset vector if o is the difference between the centers of gravity of the two 3D point sets.

Let us subtract the center of gravity from the datasets. Let denote the i^{th} points of the new point sets by a'_i and b'_i , respectively. The registration error is modified as follows: $\sum_{i=1}^N (a'_i - sRb'_i)^T (a'_i - sRb'_i)$ It can be shown by calculating the derivative of the error function with respect to R that the minimization problem is equivalent to maximizing the $\sum_{i=1}^N sa_i'^T Rb'_i$ expression w.r.t. R . Scale vector does not influence the maximum, thus the problem is to maximize $\sum_{i=1}^N a_i'^T Rb'_i$. Arun et al. ¹ proved that if the SVD of matrix $\sum_{i=1}^N b'_i a_i'^T$ is UDV^T , then the optimal solution is $R = VU^T$.

Let us rotate the second point set by R . The rotated vectors are denoted by double prime: $b''_i = Rb'_i$. The registration error becomes $\sum_{i=1}^N (a'_i - sb''_i)^T (a'_i - sb''_i)$. The scale factor can be calculated optimally by differentiating the registration error with respect to the scale: $\sum_{i=1}^N (sb''_i^T b''_i - a_i'^T b''_i) = 0$ The solution is given by the following formula: $s = \sum_{i=1}^N a_i'^T b''_i / \sum_{i=1}^N b''_i^T b''_i$. Note that Horn et al. ⁸ also proposed a formula to compute the scale, but their solution is not optimal.

Accuracy Analysis of an Enhanced Peak Detector

Tibor Kovács, Department of Automation and Applied Informatics, Budapest University of Technology and Economics

Abstract

A novel peak detection method is discussed in this paper applied for an active triangulation laserscanner. The peak detector is used to find the initial point of a line tracker algorithm. The focus of the development was on the accuracy. Six other widely used peak detectors are compared to the discussed one from this viewpoint.

First part of the paper describes the motivation of the development and summarizes previous results. The second section presents the methodology of the development. Third section describes the defined peak detector. In the fourth part the six concurrent peak detectors are presented and compared to the defined algorithm. Finally the conclusion and some application examples are demonstrated.

1. Motivation

The research touched in this paper was started when computer graphics in Hungary was low represented in electronic media. The accuracy, safety and efficiency of the industrial production was not widely enhanced by image processing systems. Nevertheless, it could be seen that this field of computer science was developing dynamically.

Industrial branches absorbing scientific results induced more research and innovation. At first this advancing effect was considerable in visual culture, because the advertisement market was more potent than industry. More and more films, titles, multimedia publications produced by animation and digital modeling technology were appearing. In the film industry a competition was evolved for the more and more dazzling, compelling or often invisible video effects published by a computer graphics studio. From this time forth it was not only a visual game; more and more news could be read about budget, income figures or success of a production from Hollywood. Today an impressive infrastructure is built behind these results: hardware elements, theoretical and practical knowledge, and software technology.

The development was started in this environment. First it was exciting to create non-existing worlds on the screen simply based on the laws of fantasy. However, when synthetic scenes could not be enhanced significantly by tuning of parametric mathematical models, demand (and the possibility as well) was arisen to get the reality and convert it to the language of computer. Systems are existing to extract geometry, surface texture, colour, even the movement of objects. Furthermore, these systems depending on configuration serve not only the entertainment industry but help in medical tasks, installed in the sensing systems of industrial robots, safety or remote sensing systems, in some cases help sportsmen or people with handicap.

Unfortunately, this technology is generally expensive. Because of this reason one of the targets of the research was to apply cheaper hardware tools and configuration, at the same time software solutions should reach as high accuracy of 3D CAD model creation as possible. After some initial success it was

realized that the noise of the workflow degrades the accuracy of the resulting model. The research target was not to enhance the accuracy by installing more advanced hardware elements (lower noise, higher resolution) the statistical integration of the noise became focused more and more. Naturally, research and development is not shut down and it is intended to apply the device in the education or can be a good basis of further R/D projects.

The intention of my study was to create a low-cost, 3D scanning system with clear structure. The designed system is based on the active triangulation principle with line-structured light source. During the development accuracy was in focus, any other circumstances – including the speed of the system – were secondary. The subject of the research was to control, handle geometrical differences between real and digital model by software solutions, and how the noise originating from different sources can be managed.

2. Configuration

The layout of the experimental device includes a light source, an imaging system, an object table and a computer. The system is presented in Figure 1.

Object table: The object is set onto a rotating table in order to creating vertical sections of the object. The table is driven by a stepping motor connected to the table item by an XL timing belt. The gear ratio is approx. 1:20, the minimal step of the motor is 1.8 degree, so the table can be rotated by 0.05 degree increment. The control of the motor is designed based on an ATME1 microcontroller that handles coils of the motor and the reset sensor (an infra gate).

The firmware of the controlling card receives commands from the host computer via serial connection. An elemental rotating process is divided to three parts because of the safe and accurate rotating of an object: an acceleration phase, constant velocity phase, deceleration phase.

The table can be set to the initial position with the signal of the reset sensor. It is important, because the camera calibration object

has to be fixed onto the table, and during the calibration it has to be located and oriented in a well-defined position.

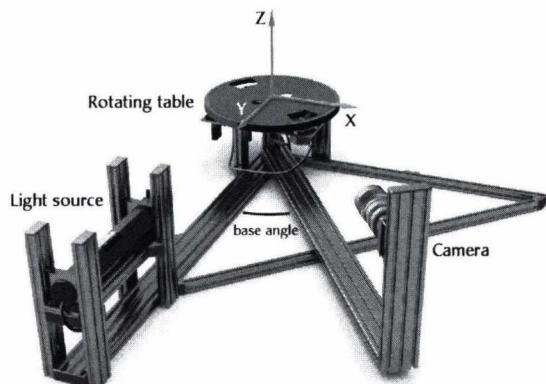


Figure 1: The configuration of the active triangulation scanner, with rotational table, laser source and imaging system.

The table can be set to the initial position with the signal of the reset sensor. It is important, because the camera calibration object has to be fixed onto the table, and during the calibration it has to be located and oriented in a well-defined position.

It is planned to build a sliding object table as well in medium term (Figure 2.). The application using this hardware is prepared to handle transformations in connection with translational scanning method.

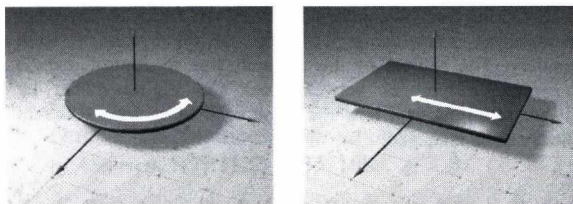


Figure 2: Rotating table for creating cylindrical sections and sliding table for producing linearly distributed sections.

Light source: The light source is a He-Ne laser with additional optics (Harry [1979]). The light beam is to be dispersed to a vertical line by a cylindrical lens. The quality of the line is enhanced by a collimating system constructed by two lenses with focal length of 37.2 mm shown in Figure 3. A temporal coherence reducing element can be inserted into this optical unit in order to deflate the disturbance of the speckle noise characterizing laser systems. This kind of device is the reduced speckle noise light source of Lasiris (StockerYale [2009]).

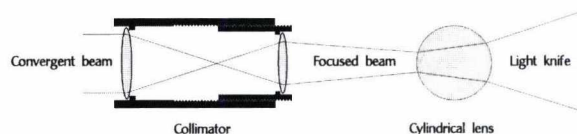


Figure 3: The collimator and the light knife.

Image Sensor: The image sensor is a Lumenera 130C camera head connected via USB to the host computer (Lumenera [2009]). The CCD has resolution of 1.3 megapixel. The system is ideal for programming because a lot of low-level functions can be reached and the native pixel flow can be got from the CCD. A C++ based driver was created with a C# interface, because the host application is coded in C#.NET environment.

Zoom, focus and aperture can be set manually.

Frame: The frame is constructed of aluminium alloy partially based on ITEM type standard elements. The mechanical design is focused on the freedom of the development. So the position of light source, camera can be set, the base angle of the triangulation can be defined coaxially with the rotating axis of the table. In order to reduce the unwanted reflections by the elements of the frame all of the element will be eloxed to black.

Software: The low level parts of the driver application cooperating with the scanner is developed in C++. Camera driver and the serial port controller of the motor are encoded in this way. The higher level functionality of the system is implemented in .NET C#. The visualisation module is constructed using DirectX.

The main application is running under Windows XP operation system.

3. Accurate tracking of a line

Results are pointing at the expected accuracy of the generated geometrical model based on probability theory, taking into consideration circumstances, conditions and algorithms shown in related publications. Parameter set can be calculated under given accuracy conditions, as well.

The line tracking method consists of two essential steps. The first step is a peak detector to find the initial point of the line tracking with subpixel resolution, applying real coordinate values instead of integers, in order to reach the maximum accuracy. This phase is the subject of present paper. A comparison is given between some widely applied peak detectors from the viewpoint of accuracy.

In the second step an algorithm is defined for the line tracking, focusing on the direction finder starting from intensity maximum. The line tracking is started in two directions from the initial point: upward and downward. More detailed information about the line tracking can be found in (Kovács [2007]).

In order to guarantee the mathematical clarity of theoretical results a research framework was defined to prove and confirm statements by measurements.

4. Methodology of research, application

The development has run on two branches. On the one hand new scientific results was examined, tuned, analysed and documented in a simulation framework. It was necessary because

of the specialities of this field. In some cases decomposition or superposition of signals occur that cannot be studied themselves in a real environment. For instance a real video signal cannot be decomposed to a noiseless signal and a neutral noise. The bridge between the real noisy environment and theoretical algorithms is a noise model that can be used by the algorithms of the synthetic environment. The validity of this noise model is proven in the built configuration with the installed image sensor. A measurement environment was designed and built that is described in (Kovács [2007]).

On the other hand, the designed 3D scanner device was created with cooperation of my fellows and students. This system validates the elaborated procedures on real objects and it serves as a good basis of further research or using the aggregated knowledge in the education. To meet this ideal goal the device was designed and built of open hardware and software elements. It means that changing or adding items in software or hardware modules are easy and standardized.

The simulation framework was implemented in Visual Studio C# and Microsoft Excel - Visual Basic. The driver system and user interface was developed in Visual Studio .NET/C# system with support of Direct3D and C++.

5. Finding the initial point: definition of the peak detector

The intensity profile of the light source in a scan line of the frame buffer is Gaussian (Tradowsky [1971]), and noise originated from the digitizing process is superimposed onto it. A number of effects distort the ideal signal in the system. For instance the geometrical distortion of the imaging system, the non-ideal features of the laser source and the optical elements, the distortions originated from the nonideal surface characteristics of the workpiece (unwanted reflections, errors generated by the roughness or texture of the object surface) and the noise of the imaging system. The image consists of two effects:

- The noiseless signal of the laser line with Gaussian intensity cross section changing the deviation and the amplitude depending on the geometry and optical features.
- Noise superimposed onto this ideal signal originating from a number of sources. This noise is additive, zero mean. There is no restriction on the distribution of the noise.

The task is to find the symmetry point of this noisy profile along a scan line. This point will be the initial point of the line tracking. The finding process is taken in real domain in the image space. The result (*u* coordinate of the initial point of the line tracking) is real, so this is a subpixel accuracy procedure.

Definition of EGA (Enhanced Gauss Approximation) algorithm: The initial point of the line tracking can be determined in the image space defined above along a sample set of a scan line as the symmetry point of the Gaussian regression. The goodness of fit is characterised by the minimum of the square error between the noisy profile and the regression curve. The initial point *u* coordinate is calculated as:

$$m = \frac{\sum_{i=1}^n (x_i - \bar{x}) \ln y_i}{2 \sum_{i=1}^n (x_i - \bar{x})^2} \cdot \frac{\left(\sum_{i=1}^n (x_i - \bar{x})^2 \right)^2 - n \sum_{i=1}^n (x_i - \bar{x})^4}{n \sum_{i=1}^n (x_i - \bar{x})^2 \ln y_i - \sum_{i=1}^n \ln y_i \cdot \sum_{i=1}^n (x_i - \bar{x})^2} + \bar{x}$$

where *m* is the real *u* coordinate of the initial point, *x_i* is the *i*th coordinate of the *i*th sample element, \bar{x} is the average of them, *y_i* is the intensity value of the *i*th sample element, *n* is the length of the sample.

The formula is originated from the following idea. A parabolic regression has to be made on the logarithm of the pattern. The function to fit:

$$y_i = f(x_i) = A \exp \frac{-(x_i - m)^2}{B}$$

After the ln operation:

$$z_i = g(x_i) = \ln f(x_i) = \ln y_i = \ln A - \frac{x_i^2 + m^2 - 2mx_i}{B}$$

The formula used for the regression:

$$g(x_i) = \beta_0 + \beta_1 x_i + \beta_2 x_i^2$$

Written the normal equations the EGA formula can be derived as a solution.

Figure 4. presents the operation of the EGA peak detector in simulation environment, figure 5. demonstrates a result in real environment.

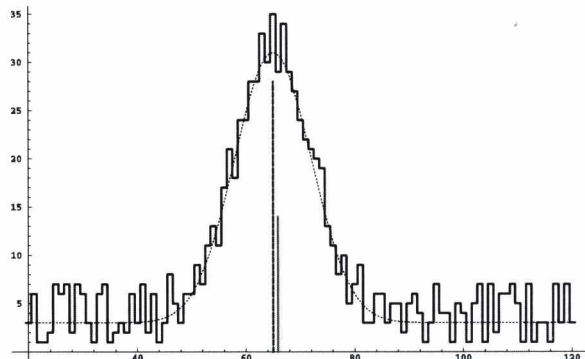


Figure 4: Working of the EGA peak detector in artificial environment.

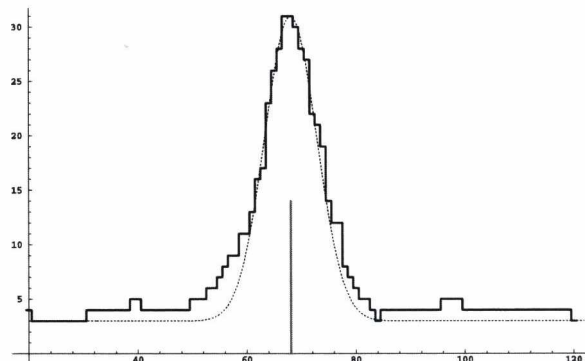


Figure 5: Working of the EGA peak detector in real environment.

It is proven by measurement in the next section, that the **EGA** procedure is more accurate than six widely used peak detectors.

6. Comparison

Forest, Salvi, Cabruja and Pous published a comparative analysis about peak detectors (Forest: Laser... [2004], more detailed: Forest: New Methods... [2004]). In their work a study was referred from Fisher and Naidu (Fisher [1996]). In this publication the five widely applied peak detectors are compared from the viewpoint of accuracy and noise sensitivity. Henceforth Forest et al. recommended a sixth one, that is more effective in noisy environment than the previously introduced five algorithm. The five detectors are the following:

- Gaussian approximation (**GA**)

$$\delta = \frac{1}{2} \cdot \left(\frac{\ln(a) - \ln(c)}{\ln(a) + \ln(c) - 2 \cdot \ln(b)} \right)$$

- Centre of mass (**CM**)

$$\delta = \frac{c - a}{a + b + c}$$

- Linear approximation (**LA**)

$$X = \begin{cases} c > a : x - \frac{a - c}{2(b - a)} \\ c \leq a : x - \frac{a - c}{2(b - c)} \end{cases}$$

- Blais and Rioux detector (**BR**)

$$\delta = \begin{cases} f(i+1) > f(i-1) : \frac{g(i)}{g(i) - g(i+1)} \\ f(i+1) < f(i-1) : \frac{g(i-1)}{g(i-1) - g(i)} \end{cases}$$

- Parabolic estimator (**PE**)

$$\delta = \frac{1}{2} \cdot \left(\frac{a - b}{c - 2b + a} \right)$$

where δ stand for the subpixel offset, the a , b and c stand for the three consecutive pixels of the peak, where b is the maximum in intensity value, $f(n)$ is the image function, $g(n)$ is the filtered derivative of the image function, X is a real position coordinate.

The sixth algorithm defined by Forest et al. (**FA**) is a combination of a noise reduction by a convolution and a first order differential line detector. Since the laser knife has Gaussian profile, the task is to find the zero crossing of the first derivative of this profile, it represents the maximum place of the cross section.

Forest approximation (**FA**)

$$X = x_0 - \frac{y_0 \cdot (x_1 - x_0)}{y_1 - y_0}$$

where x_0 is the position of the pixel preceding the zero-crossing position, x_1 is the position of the pixel after the zero crossing, y_0 and y_1 are intensities of these pixels respectively.

Since the referred source was not detailed enough to reconstruct measurement circumstances, I have implemented the **EGA** and the **FA** (the best procedure recommended by Forest et al.) algorithm in one framework for the analysis of the accuracy. Thus both procedures could be run in the same test environment with a number of settings. In noiseless case both of them calculated the position of the maximum with 100% accuracy under

0.05 pixel resolution. The changed parameters during the measurement were the following:

- Gaussian profile line width at half amplitude (2, 4)
- Deviation of the noise (zero mean normal distribution noise with deviation 0.01, 0.02, 0.03)
- Size of the convolution kernel of the **FA** algorithm (1 (no noise reduction), 3, 5, 7 window size)
- Noise threshold of **EGA** algorithm (0.3, 0.5, 0.7)

An item of the test sequence was the following. A scanline was generated with an ideal noiseless Gaussian profile loaded by the noise. The maximum position of the profile has been run in domain $x=10..40$, with 0.05 pixel steps. Thus one item of the sequence consisted of 601 measurements. Four parameters above was changed in sequences, thus 60 test sequences was generated. The absolute error and deviation of results of **EGA** and **FA** algorithms were highlighted in sequences. The result is presented visually in Figure 6.

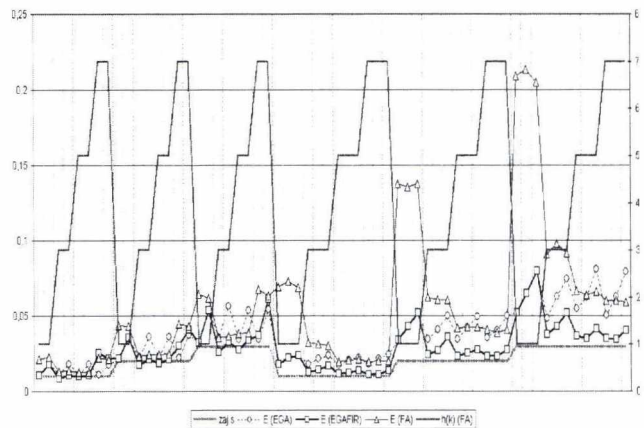


Figure 6: Comparison of the absolute error of EGA, EGAFIR, FA

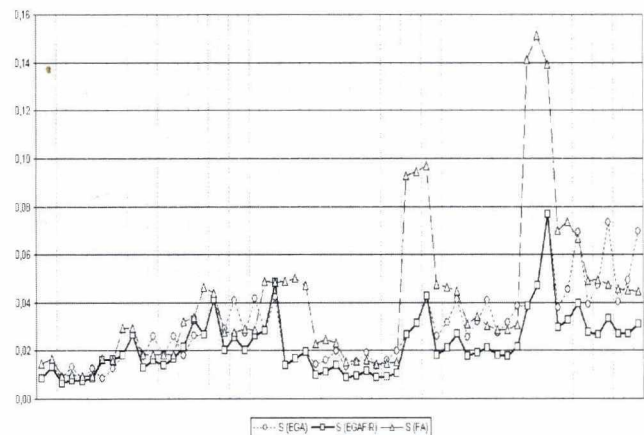


Figure 7: Comparison of the deviation of error of EGA, EGAFIR, FA.

In summary, the FA produces weaker absolute error in 76% of the cases than the EGA method in spite of applying FIR filtering. If the same FIR noise reduction is inserted into EGA procedure than in the FA (EGAFIR), this figure is improving to 96%. In my experiences the FA is very sensitive to the noise, that can be explained by the derivative procedure. The EGA is more balanced taking into consideration the whole measurement. It is presented in Figure 7.

It is interesting that there is an algorithm based on Gaussian Approximation among peak detectors published by Forest team (GA). GA uses three consecutive samples (the central is maximal). The EGA method takes into consideration all samples of profile over a threshold. Naturally the EGA approach demands more calculations, but based on the results it is more accurate. On the other hand, the EGA method can calculate the line width and the amplitude if it is necessary opposite to other peak detectors.

7. Conclusion

A novel peak detection method is defined in this paper. The peak detector is a component of the workflow of a laser scanner based on the active triangulation phenomenon built in our department.

The line tracking subsystem of the laser scanner consists of two steps. The first is finding an initial point of the line following procedure, the second is the line following itself. In the paper the **Enhanced Gaussian Approximation (EGA)** algorithm is defined and analysed for the former step.

The peak detector works alongside a scanline for the search of the intensity peak of the laser knife having a Gaussian cross-section with a subpixel accuracy. The algorithm has been compared for accuracy to six widely used peak detectors. It is proven by measurement that the Enhanced Gauss Approximation (EGA) procedure is the most accurate among the studied procedures. The absolute error of the peak detection with the EGA method was smallest in the 76% of measurement cases. If a FIR Filtering is applied for the EGA, similar to the best of six reference peak detectors (FA: Forest Approximation), the EGA is the best in the 96% of measurement cases.

The statistical proof is generated in a measurement framework programmed in MS Excel/Visual Basic and .NET C#. Results are used in the scanner system.

References

1. Harry, John E. [1979]. „Industrial Lasers and Applications” (Ipari lézerek és alkalmazásuk, in Hungarian), Műszaki Könyvkiadó.
2. StockerYale Inc.: www.stockeryale.com/lasers
3. Lumenera Corporation: www.lumenera.com
4. Tradowsky, Klaus [1971]. „A Laser ABC-je”, Műszaki Könyvkiadó.
5. Forest, Josep., Joaquim Salvi, Enric Cabruja, Carles Pous [2004]. „Laser stripe peak detector for 3D scanners. A FIR filter approach”, Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004., Volume 3, pp. 646-649.
6. Forest, Josep Collado [2004]. „New Methods for Triangulation-based Shape Acquisition Using Laser Scanners”, Tesi Doctoral, ISBN: 84-689-3091-1, p. 150.
7. Fisher, R. B., D. K. Naidu [1996]. „A Comparison of Algorithms for Subpixel Peak Detection”, in „Image Technology, Advances in Image Processing, Multimedia and Machine Vision”, Springer-Verlag, Sanz, Jorge L.C. (Ed.), pp. 385-404.
8. Kovács Tibor [2007]. „Accuracy of a Line Following Method in Noisy Environment Based on a Measured Noise Model”, GRAFGEO 2007: IV. Magyar Számítógépes Grafika és Geometria Konferencia. Budapest, Magyarország, 2007.11.13-2007.11.14.

Accurate 3D Reconstruction and Camera Auto-Calibration

Ákos Pernek, Levente Hajder

Computer and Automation Research Institute
Hungarian Academy of Sciences

Department of Automation and Applied Informatics
Budapest University of Technology and Economics

Abstract

Motion-based 3D reconstruction (SfM) with missing data has been a challenging computer vision task since the late 90s. Under perspective camera model, one of the most difficult tasks is camera auto-calibration which determines intrinsic camera parameters without using any known calibration object or assuming special properties of the scene.

This paper presents a novel algorithm to perform camera auto-calibration benefiting from multiple images and dealing with the missing data problem. The method supposes semi-calibrated cameras (every intrinsic camera parameter except for the focal length is considered to be known) and constant focal length over all the images. The solution requires at least one image pair having at least eight common measured points.

Tests verified that the algorithm is numerically stable and produces accurate results. Based on the obtained camera calibration data, most reconstruction methods using perspective camera model are able to reconstruct the 3D structure of the object, thus the technique is of great importance for the SfM problem.

Categories and Subject Descriptors (according to ACM CCS): I.4.1 [Digitization and Image Capture]: Camera calibration

1. Overview

Camera calibration is the method to obtain intrinsic camera parameters. In the past years, a number of different camera auto-calibration methods have been published. The first approach was presented in ^{3,4}. The paper proposed a camera 'self-calibration' method which was able to perform camera calibration without using a known calibration object. The method utilized the Kruppa equations which link epipolar transformation to the image of the absolute conic. From at least three camera movements the Kruppa equations can be solved for the image of the absolute conic and the intrinsic camera parameters can be obtained. The drawback of this approach is that it requires non-linear solution. An alternative way is stratification ^{5,7}. First affine structure is recovered and it is used to solve for the calibration linearly. Another approach is to explicitly locate the absolute quadric in an initial projective reconstruction and use it to 'straighten' the projective structure ⁶. A large number of algorithms have been published on the basis of these previous approaches with the

application of different constraints (e.g. constant focal length over all the images). Camera auto-calibration methods using the Gröbner basis theory ¹⁵, and the hidden variable technique ⁹ have been published. These methods could solve the self-calibration problem from a minimal set of points and two views. The current paper focuses on the multi-image extension of the method ¹³ which provides a simple transformation of Gröbner basis solution on multivariate polynomial equation sets to generalized eigenvalue problem class and therefore greatly simplifies the solution.

In computer vision it is essential to determine intrinsic camera parameters. Having calibrated cameras is fundamental in many research areas. One of these areas is the well-known structure from motion (SfM) problem. The aim of SfM is to reconstruct 3D structure based on pictures of the original object(s) taken from multiple different camera positions and orientations.

This study introduces a novel camera auto-calibration method which benefits from multiple image measurements

and handles the missing data problem. The method assumes all camera parameters but the focal length to be known and fixed focal length over all the images. The algorithm is formulated as a rectangular generalized eigenvalue problem and refines its solution in an alternating fashion providing accurate results.

2. Fundamental constraints

This section introduces important constraints widely used for structure from motion and camera auto-calibration applications as well.

Introduced constraints apply to image point correspondences which are obtained by tracking feature points. The 3D points of an object project to the image plane of the taken images. A number of these points (features) can be identified and tracked along these images. The motion of a single 3D point defines a motion trajectory. Measured point coordinates on motion trajectories are called point correspondences.

Consider a stereo image pair with two calibrated views and known point correspondences. As defined in ¹⁶, the epipolar (1) and the determinant (2) constraints can be written:

$$x_j^{i'T} F x_j^i = 0 \quad (1)$$

$$\det(F) = 0 \quad (2)$$

where x_j^i and x_j^i denote corresponding point coordinates on the image planes of the image pair. F denotes the so-called fundamental matrix encoding epipolar geometry ¹⁶. At least eight independent points are sufficient to recover a unique fundamental matrix. The application of constraint (2) enables us to reduce the necessary number of points to seven, however, solution in general is not unique. Applying the third (5) constraint (introduced later) and assuming a semi-calibrated camera makes it possible to recover the fundamental matrix from only six point correspondences, however, the solution in general will not be unique again.

This paper focuses on perspective camera model ¹⁶. Let C and C' denote the camera calibration matrices defining internal camera parameters. Considering the correlation between points in the coordinate system of the camera and measured points in the image planes, the following equations can be written:

$$x_j^i \simeq C x_j^c, \quad x_j^i \simeq C' x_j^{c'} \quad (3)$$

where $(x_j^c, x_j^{c'})$ denotes the 3D points in the camera coordinate system while (x_j^i, x_j^i) denotes the 3D homogenous coordinates on the image plane. Substituting (3) into Equation (1) leads to the following formula:

$$x_j^{i'T} C'^{-T} E C^{-1} x_j^i = 0 \quad (4)$$

The fundamental matrix encapsulating the intrinsic geometry between the images can be written as $F = C'^{-T} E C^{-1}$. Here E is a 3 x 3 rank-2 matrix called the essential matrix.

The essential matrix contains the rotation (R) and translation (t) which transform a camera of the stereo image pair into the other camera ($E = [t]_x R$) ¹⁶. A property of the essential matrix is that its two non-zero eigenvalues are the same. This property is the basis of the trace constraint:

$$2EE^T E - \text{trace}(EE^T)E = 0 \quad (5)$$

Substituting $E = C'^T F C$ derived from $F = C'^{-T} E C^{-1}$ into the trace constraint (5) gives the trace constraint for the fundamental matrix:

$$2F Q F^T Q F - \text{trace}(F Q F^T Q) F = 0 \quad (6)$$

where $Q = C C^T$ and $C = C'$ (fixed internal camera parameters assumed) and noting that camera matrix can be written as $C \simeq \text{diag}([1, 1, \frac{1}{f}])$.

With calibrated cameras, the essential matrix can be derived from the fundamental matrix and the transformation is invertible. Then the essential matrix can be used to acquire relative pose and orientation between the cameras of the stereo image pair. In case of calibrated views, this information is sufficient to perform a structure reconstruction based on the measured image points.

The next sections investigate how cameras can be calibrated from measured points using the constraints introduced above.

3. Generalized eigenvalue problem

To perform camera calibration utilizing constraints defined in Section 2, the necessity of solving a set of multivariate matrix polynomial equations emerges. Having semi-calibrated cameras the number of variables reduces to one, but the task to solve a redundant, quadric equation set still exists. The problem of solving an equation set of this kind is a challenging task for numerical polynomial algebra.

In the past, a number of studies have been published on SfM related problems applying different kinds of solutions to multivariate matrix polynomial equations. These papers mostly aimed to solve famous minimal problems like the 5-pt relative pose or the 6-pt unknown focal length problem. The two most widely used methods were the hidden variable technique and the Gröbner basis method. The hidden variable technique is presented in ⁹ and ¹⁰ for both the 5-point relative pose and the 6-point unknown focal length problem. The current state-of-the-art methods for the 5-pt relative pose problem are presented in ¹¹ and ¹². The latter is based on the Gröbner basis method.

The Gröbner basis method has proven to be applicable in many areas of science, however, in ¹³ a new approach was presented. The coefficient matrix of the multivariate polynomial equations is transformed into a form realizing a new problem class, the so-called polynomial eigenvalue problem class.

Using similar notations as in ¹³, the polynomial eigenvalue problem (PEP) is introduced here as well. Consider an equation $A(\lambda)x = 0$ where the matrix polynomial $A(\lambda)$

is equal to $\lambda^l W_l + \lambda^{l-1} W_{l-1} + \dots + \lambda W_1 + W_0$ where W_j denote $n \times n$ square matrices. This equation can easily be transformed to a new problem class, the so-called generalized eigenvalue problem class (GEP) ⁸. The generalization of the eigenvalue problem can be written as $Av = \lambda Bv$.

Narrowing the solution to our special case, when $l = 2$, above matrix polynomial can be rewritten to the following GEP form:

$$A = \begin{pmatrix} 0 & I \\ -W_0 & -W_1 \end{pmatrix}, \quad B = \begin{pmatrix} I & 0 \\ 0 & W_2 \end{pmatrix}, \quad v = \begin{pmatrix} x \\ \lambda x \end{pmatrix} \quad (7)$$

GEP is mathematically well studied and a number of efficient algorithms are available to solve it. However, the number of solutions compared to the original problem is increased from n to nl .

The next sections explain how GEP is used to support camera auto-calibration.

4. The proposed method: multi-image polynomial eigenvalue solver (MPEig)

This section explains the proposed algorithm: *MPEig* in details. To have a short overview on the *MPEig* method, an algorithm summary is presented below which shows the high level steps of the method.

Algorithm 1 Summary of the MPEig algorithm

```

 $\tilde{W} \leftarrow \text{Init}(\text{mask}, W, \text{image width}, \text{image height})$ 
for all image pairs do
     $Fs(i) \leftarrow \text{ComputeFundamental}(\text{mask}, \tilde{W})$ 
end for
 $\tilde{F}s \leftarrow \text{FilterFundamentals}(Fs)$ 
for all  $\tilde{F}$  fundamentals in  $\tilde{F}s$  do
     $W_2^i, W_1^i, W_0^i \leftarrow \text{ComputeCoeffMatrices}(\tilde{F})$ 
end for
 $W_2^{\text{rect}}, W_1^{\text{rect}}, W_0^{\text{rect}} \leftarrow \text{AssembleCoeffMatrices}(W_2^i, W_1^i, W_0^i)$ 
 $A, B \leftarrow \text{ConvertToRectGEP}(W_2^{\text{rect}}, W_1^{\text{rect}}, W_0^{\text{rect}})$ 
 $fs \leftarrow \text{SolveRectGEPForMultipleImages}(A, B)$ 
 $f \leftarrow \text{ChooseFocal}(fs)$ 

```

4.1. Initialization

The *MPEig* algorithm assumes fixed focal length (f) and semi-calibrated cameras with zero skew (s), one image aspect ratio (r) and known principal point (u_0, v_0) located at the center of the image plane (*image width* / 2, *image height* / 2). These assumptions are common and therefore they can be made without reducing the applicability of a certain method. The first step of the algorithm is to modify measured point correspondences (W) so that they fulfill the assumptions. Camera calibration matrices (C) are having form: $[f \ s \ u_0; 0 \ -rf \ v_0; 0 \ 0 \ 1]$, therefore the modification is done via multiplication of a matrix: $[1 \ 0 \ -u_0; 0 \ -\frac{1}{f} \ \frac{1}{f}v_0; 0 \ 0 \ 1]$ changing the coordinate system so that the origin is moved from the image center (u_0, v_0) to $(0, 0)$ and the aspect ratio (r) is changed to one.

4.2. Computing fundamental matrices

The method computes the fundamental matrices for every image pair from at least eight points. A great variety of fundamental matrix estimators can be used to perform this computation. A number of efficient algorithms can be found in ^{18, 22, 1}. For practical reasons we use the method of Hartley ¹.

MPEig also handles the missing data problem. Due to camera motion and object shape, it is a usual situation that the tracked feature points are not visible on every images. The visibility of measured points are modeled with a visibility matrix (*mask*). For computation of the fundamental matrices only visible points are utilized. Once fundamental matrices are computed, the missing data problem is not an issue any more for the proposed method.

4.3. Filtering fundamental matrices

Due to degeneracy of measurements in presence of noise, it is possible to acquire imaginary, infinite or not-a-number fundamental matrices. The algorithm simply drops invalid fundamentals.

4.4. Computing coefficient matrices

Having valid fundamental matrices, the coefficient matrices of the PEP form are derived. Equation (6) can be rewritten into matrix polynomial form for each fundamental matrix: $w^2 W_2^i + w W_1^i + W_0^i = 0$ where w denotes $\frac{1}{f^2}$ and the coefficient matrices W_2^i, W_1^i and W_0^i are of size 3×3 . For detailed computations refer to Appendix A.

The above equations can be solved as GEP for each image pair after transformation to the proper form as described in Section 3. However, *MPEig* solves all the equations together.

4.5. Assembling coefficient matrices

The W_2^i, W_1^i and W_0^i coefficient matrices can be assembled together to form rectangular matrices: $W_2^{\text{rect}}, W_1^{\text{rect}}$ and W_0^{rect} . See Equation (11) for detailed form. *MPEig* algorithm operates on these rectangular matrices making it possible to use data from multiple measurements.

4.6. Conversion to rectangular GEP form

The GEP solution explained in ¹³ simplifies computations, however, it has certain limitations from the aspect of multi-image usability. The fundamental matrix is computed from only six points. The original fundamental matrix is rewritten as the components of the null-space vectors for the epipolar constraint ($F = xF_1 + yF_2 + F_3$). This parameterization is then substituted into the rank constraint (2) and the trace constraint for the fundamental matrix (6) giving ten third order polynomial equations in unknowns x, y and $w = \frac{1}{f^2}$. Then

$\lambda = w$ choice provides the PEP form:

$$(w^2 C_2 + w C_1 + C_0)v = 0 \quad (8)$$

Equation (8) is a special case of PEP. It is called QEP (quadratic eigenvalue problem). The QEP of course can be solved with GEP. The solution is a number of $w = \lambda$ eigenvalues directly providing unknown camera focal lengths based on $w = \frac{1}{f^2}$ equation. The corresponding eigenvectors (v) are solved for x, y and therefore they provide the fundamental matrices belonging to the camera focal length derived from the current w eigenvalue. This method is numerically stable and generates good results in case of a stereo image pair. However, as the fundamental matrices serve as eigenvectors in the solution, the method is not applicable for multiple images because different image pairs has different fundamentals, so it is not possible to have a common eigenvector for all image pairs.

In this paper the fundamental matrices are computed from at least eight points uniquely and they are directly substituted into the trace constraint for the fundamental matrix (6). The advantage of this solution is that the eigenvector v corresponding to the desired solution w is common for all images and it is independent from the fundamental matrices. It provides the basis of the rectangular GEP formalization and the basis of the solution utilizing multiple-image measurements (see the next section for details).

4.7. Multi-image solution

Using more data to obtain camera focal length increases accuracy and numerical stability. This leads to the idea to extend the proposed method to handle the multiple image case. This section shows how the algorithm is extended.

Shortly summarizing the previous sections, the algorithm is the following: suppose that m images are available of the object. In this case $n = \frac{m(m-1)}{2}$ image pairs exist. If the i^{th} image pair has at least eight common measured points, the i^{th} fundamental matrix $Fs(i)$ can be computed. Let W_2^i, W_1^i , and W_0^i denote the coefficient matrices belonging to this fundamental matrix as defined in Section 4.4. The coefficient matrices can be computed for all valid image pairs and the following rectangular generalized eigenvalue problem can be acquired:

$$A^{rect} v = \lambda B^{rect} v \quad (9)$$

where

$$A^{rect} = \begin{pmatrix} 0 & I \\ -W_0^{rect} & -W_1^{rect} \end{pmatrix}, \quad B^{rect} = \begin{pmatrix} I & 0 \\ 0 & W_2^{rect} \end{pmatrix} \quad (10)$$

W_2^{rect}, W_1^{rect} , and W_0^{rect} matrices are the cumulate coefficient matrices composed of coefficient matrices derived from all valid fundamental matrices:

$$W_j^{rect} = (w_j^1, w_j^2, \dots, w_j^f)^T, \quad j = 0..2 \quad (11)$$

The problem with Formula (9) is that A^{rect} and B^{rect} are non-square matrices, therefore the generalized eigenvalue solution cannot be applied directly. However, a number of solutions exist to solve the rectangular generalized eigenvalue

problem as well. An efficient method is proposed in ¹⁴. It computes initial estimates for w and then refines them in an alternating fashion. It consists of two steps as follows:

1. Initialization

It has been proven that Equation (9) can be transformed to a lower dimension by left-side multiplication with the transpose of B^{rect} (for convenience, transpose is denoted with a prime). This transformation reduces the size of matrices A^{rect} and B^{rect} to 3×3 so the problem is transformed back to the well-known square generalized eigenvalue problem. In noisy case this transformation can be used to estimate the noise-free 3×3 coefficient matrices. The initial estimate of w is computed using these matrices.

$$B'^{rect} A^{rect} v = \lambda B'^{rect} B^{rect} v \quad (12)$$

2. Alternation

The v eigenvector corresponding to w is then computed as the right singular vector corresponding to the smallest singular value of $(A^{rect} - wB'^{rect})^T (A^{rect} - wB'^{rect})$.

Based on the v eigenvector w can be refined according to the scalar quadric equation: $v^T (A^{rect} + wB'^{rect})(A^{rect} - wB'^{rect})v$. As $w = \frac{1}{f^2}$, w must be set to the positive root of the equation ¹⁴.

Repeating the above steps in an alternating fashion refines w step by step. The algorithm has been proven to converge. The termination criterion is that the modification of w drops below a given limit or a predefined number of iterations is reached. Achieving this, w is accepted as a valid solution, the unknown focal length of the camera can be derived from it.

The algorithm in ¹⁴ aims to find the original (noise-free) A^{rect} and B^{rect} matrices. Tests demonstrated that its application increases the accuracy of the solution. In case of *MPEig* optimization has been executed on all candidate focal lengths.

4.8. Choosing focal length

Equation $w^2 W_2^{rect} + w W_1^{rect} + W_0^{rect} = 0$ can provide at most six possible solutions on w . Therefore the proper solution has to be chosen. As w denotes $\frac{1}{f^2}$, possible valid solutions for f correspond to positive, finite and real values of w . However, in most cases this criterion is not enough to filter out all invalid solutions. All candidate w values are substituted into the polynomial matrix equation above. The w value which minimizes the expression $\|w^2 W_2^{rect} + w W_1^{rect} + W_0^{rect}\|$ is accepted and the focal length f is computed from the selected w .

5. Tests

Several experiments have been performed to study the properties of the proposed method. Tests have been executed on both synthetic and real data. The host machine was an Intel(R) Core(TM) Duo T2500 computer with 2.0GHz CPU and 4Gb of memory (note that program code was written in GNU Octave which uses only one core).

5.1. Synthetic tests

5.1.1. Generation of input data

The input (measurement matrix) is composed of motion trajectories. These trajectories are generated in the following way: (i) Random three-dimensional coordinates are generated by a zero-mean Gaussian random number generator with variance σ_{3D} . (ii) The generated 3D points are rotated by random angles and translated by random vectors within reasonable limits. (iii) A semi-calibrated camera matrix is generated with random focal length in the range of [1..1000] (iv) The transformed points are projected onto the image plane using this camera matrix. (v) Noise generated by a zero-mean Gaussian random number generator is added to the projected coordinates. Its variance is set to σ_{2D} . (vi) Finally, the measurement matrix W is formed using the resulting coordinates.

For each test case, 25 measurement matrices are generated and the methods are executed 25 times. The results shown in this section are calculated as the average of the 25 executions.

5.1.2. Rival algorithms

The proposed method (*MPEig*) has been compared to other algorithms.

As *MPEig* extends the method of Kukulova et al. ¹³, their method was chosen first. Throughout the tests *PEig6Pt* (polynomial eigenvalue 6-point) will denote the algorithm of Kukulova et al. Its implementation was downloaded from the authors' web page [†] and was slightly modified to run under GNU Octave.

Finally, the absolute quadric-based camera calibration introduced by Triggs ⁶ was selected to race against *MPEig*. In case of identical, semi-calibrated cameras the method of Triggs is suitable to obtain the absolute quadric from an overdetermined linear equation set of form $Aw = 0$ where w is built up of the elements of the absolute quadric. Then, camera calibration can be computed easily from the quadric. In the followings, this method will be called *Quadric*.

Note that linear means of Triggs' method has certain limitations ¹⁶ as basic properties of the absolute quadric are not enforced by linear solution. In general, the absolute quadric will not have rank three and may not be positive semi-definite. To ensure that the rank of the quadric is three, it can be rebuilt from its SVD decomposition by setting the smallest singular value to zero. This modification of the *Quadric* (called *QuadricRank3* from now on) is also included in our tests.

Direct constrained numerical optimization ⁶ has also been evaluated as a candidate algorithm to compute the absolute quadric. However, it has been dropped from tests due to its numerical instability at high error levels.

5.1.3. Initial projective reconstruction

Traditional camera auto-calibration methods (including the absolute quadric-based ones) require an initial projective reconstruction $\{P^i, X_j\}$. Martinec and Pajdla ¹⁷ developed a promising method able to compute the projective reconstruction even when missing data occurs. They utilized submatrices of rank four of the original measurement matrix which do not include missing elements. Linear spaces generated from the rows of these submatrices have been combined to provide constraints on the basis of the complete measurement matrix. Even though the solution involves the rescaling of the measurement matrix, the method is nearly optimal if the magnitude of the rescale coefficients is approximately similar. We have chosen this method to compute the projective reconstruction required by *Quadric* and *QuadricRank3*. Our implementation is based on the code section found on the authors' web page [‡], complete submatrices have been rescaled using the method of Sturm and Triggs ⁷ and image pairs and triplets were used to provide necessary constraints as suggested in ¹⁷.

5.1.4. Comparing MPEig to PEig6Pt

Figure 1 shows focal length accuracy of the *PEig6Pt* method and the proposed *MPEig* method with different numbers of images. Focal length accuracy is defined as $|(f_{computed} - f_{gt})/f_{gt}|$ where f_{gt} denotes the ground truth focal length while $f_{computed}$ denotes the output of the algorithm. During the tests noise level was increased from 0.0 to 10.0 percent in 1.0 percent increments. In both cases 56 point trajectories over 10 images were generated and modified with Gaussian noise. The *PEig6Pt* algorithm was applied to the first two images along with RANSAC ² (assuming an outlier ratio of 50 percent) to find the most suitable six point correspondences. The candidate focal length producing the lowest focal error was chosen each time.

The result shows that *MPEig* outperforms *PEig6Pt* even in the two-image case due to its iterative refinement and robust computation of the fundamental matrix from 56 points. It can also be observed that the accuracy of the proposed algorithm increases with the number of images and its numerical stability is consistently superior. However, such comparison of *PEig6Pt* and *MPEig* is misleading as *PEig6Pt* solves a minimal problem while *MPEig* has been designed to be a robust solver. Nevertheless, it is shown that *MPEig* is a valuable upgrade of *PEig6Pt*.

5.1.5. Comparing MPEig to Quadrics

Figure 2 displays the results w.r.t. noise level which was increased from 0.0 to 10.0 in 1.0 percent increments. Number of points and images have been fixed at 100 and 7, respectively. The test verified that *QuadricRank3* behaves better than *Quadric* method, however, *MPEig* outperforms both algorithms. The stable linear solution for the absolute quadric

[†] https://cmp.felk.cvut.cz/minimal/6_pt_relative.php

[‡] <http://cmp.felk.cvut.cz/~martid1/demoCVPR05/code>

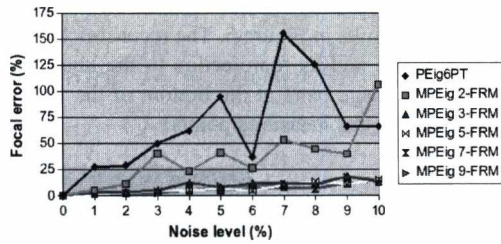


Figure 1: Focal error versus noise level

does not prove to be stable for the calibration accuracy as the properties of the quadric are not guaranteed. *MPEig* solves for only the unknown focal length and its problem formalization implicitly contains all the constraints of the camera. The running time of *MPEig* is about 4 seconds which is much greater than that of the others, however, the latter measurements do not include time demand of computing the necessary projective reconstruction that is in general time consuming.

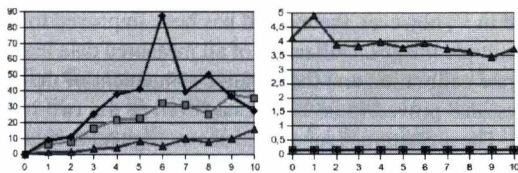


Figure 2: Percentage of focal error (left) and time demand in seconds (right) versus noise level. Notation is triangle, rhombus and square for *MPEig*, *Quadric*, *QuadricRank3*, respectively.

Figure 3 displays the results w.r.t. increasing number of points. The number of measured points was increased from 20 to 200 in increments of 20. Noise level and the number of images have been fixed at 7.0 and 7, respectively. Performance and time demand of the algorithms is similar to Figure 2, but accuracy increases with the number of points. The reason is that, in general, the more points are available, the more robust it is to compute the fundamental matrix. This is beneficial for both *MPEig* and the computation of the initial projective reconstruction. The execution time of *MPEig* increases linearly as it depends on the linear solution for fundamental matrices, while quadric methods are still very fast as their time demand is measured again without that of the projective reconstruction.

Figure 4 displays the results w.r.t. the number of frames, which was increased from 3 to 15 in increments of 2. Noise level and the number of points have been fixed at 5.0 and 56, respectively. *MPEig* proved to be numerically more stable than its rival algorithms and its running time was a slope quadratic function of the number of images as fundamental matrices are computed and used for each image pair.

Tests have been executed to examine the performance of

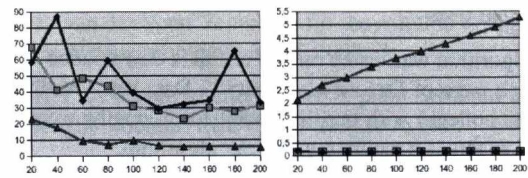


Figure 3: Percentage of focal error (left) and time demand in seconds (right) versus number of points. Notation is triangle, rhombus and square for *MPEig*, *Quadric*, *QuadricRank3*, respectively.

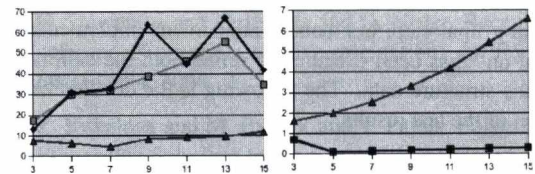


Figure 4: Percentage of focal error (left) and time demand in seconds (right) versus number of frames. Notation is triangle, rhombus and square for *MPEig*, *Quadric*, *QuadricRank3*, respectively.

the algorithm for changing level of missing data as well. During the test, the measurement matrix was fixed at 56 points and 7 images generated with 3.0 percent noise level. Tests simulated a step by step increasing amount of missing data. *MPEig* performed well, but approaching 80 percent of missing data, the results turned out to be unacceptable. For the rival methods, this behavior occurred sooner, even at about 50 percent missing data ratio. This implies, that in case of missing data, erroneous measurements have a stronger negative effect on the absolute quadric-based calibration (and on the computation of the projective reconstruction) than to *MPEig*.

5.1.6. Critical motion sequences

Finally, the method has also been tested against critical motion sequences to obtain degenerate situations according to ^{19, 20}. For stereo, *MPEig* fails when the fundamental matrix has all zeros in the diagonal (pure translation, translation along optical axis with possible rotation around the optical axis and some other special cases). There are also cases (pure rotation) when camera calibration is possible but structure reconstruction is not. When multiple images are available, the algorithm fails only when all image pairs are degenerate. Fortunately this situation is very rare. It is also possible to drop improper image pairs detecting wrong fundamental matrices using the degree of fixation indicator ²¹ for diagonal elements. However, in noisy case it is not easy to choose a proper threshold.

5.2. Real tests

The algorithm was tested on real test data downloaded from the web page of Oxford University [§]. Data sets (Wadham, Merton college) with quasi-constant focal lengths were selected as this is a requirement of our algorithm.

The auto-calibration was based on the two dimensional image measurements belonging to the data sets. Then the valid image pairs were processed by a classical stereo structure reconstruction ¹⁶ based on the previously obtained calibration data and image measurements. Finally, the substructures acquired from each stereo reconstruction were registered ² to each other.

As a final step, our bundle adjustment implementation based on ² has been executed to refine both the calibration and the structure data. The following is the detailed explanation of the test results.

1. Reconstruction of the Wadham college:

The source data for the Wadham college contains measurements in 5 images with 1331 tracked points. The missing data ratio is 55%. The calibration ran for 5.66 seconds and provided an average reprojection error of 0.008 pixels for coordinates of measured points. The application of bundle adjustment reduced this value to 0.004 in five cycles. The result can be seen on Figure 5. The reconstructed structure is very accurate, the shape is clear with well observable right angles. The recovered focal length was 1148.2 which is judged to be an accurate estimate when compared to the camera matrices derived from the given projection data. The reconstructed structure was reprojected to the image plane. Some neighboring image pairs with marked feature points computed as the reprojection of the reconstructed structure can be observed in the two image pairs at the bottom of Figure 5.

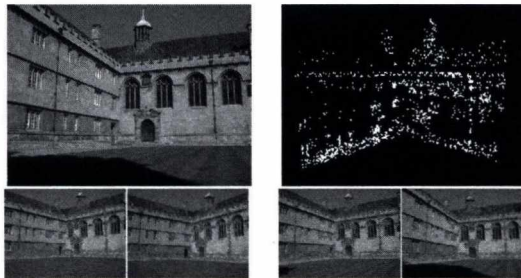


Figure 5: Reconstruction of the Wadham college

2. Reconstruction of the Merton college:

The source data for the Merton college contains measurements in 3 images with 575 tracked points. The missing data ratio is 25%. The calibration ran for 4.52 seconds and provided an average reprojection error of 0.12 pixels for coordinates of measured points. The application of

bundle adjustment reduced this value to 0.08 in five cycles. See the results in Figure 6. The reconstructed structure is accurate and its reprojection to the image plane (see the two image pairs at the bottom of the figure) is close to the original measured points.

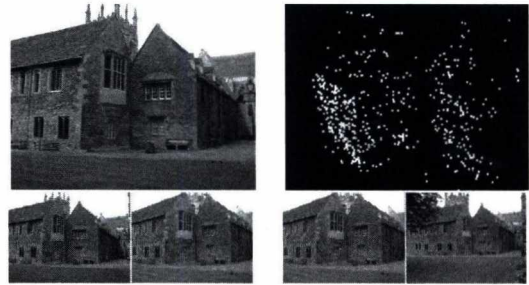


Figure 6: Reconstruction of the Merton college

6. Conclusion

In this paper a novel camera auto-calibration method has been presented. The algorithm is based on the solution published in ¹³, however, the method has been improved to handle multiple images. Utilization of all measured data leads to accurate calibration and the iterative refinement of the final results also proved to be beneficial (even in the two-image case). The algorithm is robust and shows high numerical stability. Computational time is in seconds range even for huge measurements matrices. The algorithm handles the missing data problem which is a common and in most cases an unavoidable problem for SfM applications. Finally, tests verified the applicability of the algorithm for both synthetic and real data sets. The method outperformed the rival ones.

Appendix A: Reformulation of the trace constraint

Recalling that $Q = \text{diag}([1, 1, w])$ and $w = \frac{1}{f^2}$, the trace constraint for the fundamental matrix (6) can be developed. The first step is to evaluate FQF^TQ .

$$FQF^TQ = \begin{pmatrix} f_{11} & f_{12} & wf_{13} \\ f_{21} & f_{22} & wf_{23} \\ f_{31} & f_{32} & wf_{33} \end{pmatrix} \begin{pmatrix} f_{11} & f_{21} & wf_{31} \\ f_{12} & f_{22} & wf_{32} \\ f_{13} & f_{23} & wf_{33} \end{pmatrix} =$$

$$\begin{pmatrix} f_{11}f_{11} + f_{12}f_{12} + wf_{13}f_{13} & f_{11}f_{21} + f_{12}f_{22} + wf_{13}f_{23} \\ f_{21}f_{11} + f_{22}f_{12} + wf_{23}f_{13} & f_{21}f_{21} + f_{22}f_{22} + wf_{23}f_{23} \\ f_{31}f_{11} + f_{32}f_{12} + wf_{33}f_{13} & f_{31}f_{21} + f_{32}f_{22} + wf_{33}f_{23} \end{pmatrix}$$

$$\begin{pmatrix} w(f_{11}f_{31} + f_{12}f_{32}) + w^2f_{13}f_{33} \\ w(f_{21}f_{31} + f_{22}f_{32}) + w^2f_{23}f_{33} \\ w(f_{31}f_{31} + f_{32}f_{32}) + w^2f_{33}f_{33} \end{pmatrix} = w^2A + wB + C$$

where

$$A = \begin{pmatrix} 0 & 0 & f_{13}f_{33} \\ 0 & 0 & f_{23}f_{33} \\ 0 & 0 & f_{33}f_{33} \end{pmatrix},$$

[§] <http://www.robots.ox.ac.uk/~vgg/data/data-mview.html>

$$B = \begin{pmatrix} f_{13}f_{13} & f_{13}f_{23} & f_{11}f_{31} + f_{12}f_{32} \\ f_{23}f_{13} & f_{23}f_{23} & f_{21}f_{31} + f_{22}f_{32} \\ f_{33}f_{13} & f_{33}f_{23} & f_{31}f_{31} + f_{32}f_{32} \end{pmatrix},$$

$$C = \begin{pmatrix} f_{11}f_{11} + f_{12}f_{12} & f_{11}f_{21} + f_{12}f_{22} & 0 \\ f_{21}f_{11} + f_{22}f_{12} & f_{21}f_{21} + f_{22}f_{22} & 0 \\ f_{31}f_{11} + f_{32}f_{12} & f_{31}f_{21} + f_{32}f_{22} & 0 \end{pmatrix}$$

Using this result $2FQF^TQF$ can be written into matrix polynomial form.

$$2FQF^TQF = 2w^2AF + 2wBF + 2CF \quad (13)$$

Based on the value of FQF^TQ , its trace can be calculated as a function of w .

$$\begin{aligned} \text{trace}(FQF^TQ) &= f_{11}^2 + f_{12}^2 + wf_{13}^2 + f_{21}^2 + \\ &f_{22}^2 + wf_{23}^2 + wf_{31}^2 + wf_{32}^2 + w^2f_{33}^2 = w^2 \underbrace{f_{33}^2}_{w_2} + \\ &w \underbrace{(f_{13}^2 + f_{23}^2 + f_{31}^2 + f_{32}^2)}_{w_1} + \underbrace{(f_{11}^2 + f_{12}^2 + f_{21}^2 + f_{22}^2)}_{w_0} \end{aligned} \quad (14)$$

Finally, the trace constraint can be reduced to the form of $w^2W_2 + wW_1 + W_0$ by developing (6) using (13) and (14) where $W_2 = 2AF - w_2F$, $W_1 = 2BF - w_1F$ and $W_0 = 2CF - w_0F$.

References

1. R. I. Hartley In defence of the 8-point algorithm. *International Conference on Computer Vision*, 1064-1070, 1995. 3
2. M. Fischler and R. Bolles RANdom SAMpling Consensus: a paradigm for model fitting with application to image analysis and automated cartography. *Commun. Assoc. Comp. Mach.*, 358-367, 1981. 5
3. O. D. Faugeras and Q.-T. Luong and S. J. Maybank Camera self-calibration: theory and experiments. *European Conference of Computer Vision*, 321-334, 1992. 1
4. O. D. Faugeras and S. J. Maybank A Theory of Self-Calibration of a Moving Camera. *International Journal of Computer Vision*, 123-151, 1992. 1
5. O. D. Faugeras Stratification of three-dimensional vision. *Journal of the Optical Society of America*, 12:465-484, 1995. 1
6. B. Triggs Autocalibration and the absolute quadric. *CVPR '97*, 609-614, 1997. 1, 5
7. R. I. Hartley Camera calibration and the search for infinity. *In ICCV*, 510-517, 1999. 1
8. Z. Bai and J. Demmel and J. Donggorra and A. Ruhe and H. van der Vorst Templates for the solution of algebraic eigenvalue problems. *SIAM*, 2000. 3
9. H. Li A simple solution to the six-point two-view focal-length problem. *ECCV*, 200-213, 2006. 1, 2
10. H. Li and R. I. Hartley Five-Point Motion Estimation Made Easy. *ICPR '06*, 630-633, 2006. 2
11. D. Nister An Efficient Solution to the Five-Point Relative Pose Problem. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference*, 2003. 2
12. H. Stewenius and C. Engels and D. Nister Recent developments on direct relative orientation. *ISPRS '06*, 60:284-294, 2006. 2
13. Z. Kukelova and M. Bujnak and T. Pajdla Polynomial Eigenvalue Solutions to the 5-pt and 6-pt Relative Pose Problems. *British Machine Vision Conference*, 565-574, 2008. 1, 2, 3, 5, 7
14. G. Boutry and M. Elad and G. H. Golub and P. Milanfar The generalized eigenvalue problem for nonsquare pencils using a minimal perturbation approach. *SIAM J. Matrix Anal. Appl.*, 582-601, 2005. 4
15. H. Stewenius and F. Kahl and D. Nister and F. Schaffalitzky A minimal solution for relative pose with unknown focal length. *CVPR '05*, 789-794, 2005. 1
16. R. I. Hartley and A. Zisserman Multiple View Geometry in Computer Vision. *CVPR '05*, 2003. 2, 5, 7
17. D. Martinec and T. Pajdla 3D Reconstruction by Fitting Low-Rank Matrices with Missing Data. *CVPR '05 - Volume 1*, 198-205, 2005. 5
18. K. Kanatani and Y. Sugaya Compact Fundamental Matrix Computation. *PSIVT '09: Proceedings of the 3rd Pacific Rim Symposium on Advances in Image and Video Technology*, 179-190, 2008. 3
19. P. Sturm Critical Motion Sequences for Monocular Self-Calibration and Uncalibrated Euclidean Reconstruction. *CVPR*, 1997. 6
20. F. Kahl, B. Triggs and K. Aström Critical Motions for Auto-Calibration When Some Intrinsic Parameters Can Vary. *J. Math. Imaging Vis.*, 131-146, 2000. 6
21. K. Kanatani, A. Nakatsuji and Y. Sugaya Stabilizing the Focal Length Computation for 3-D Reconstruction from Two Uncalibrated Views. *Int. J. Comput. Vision*, 109-122, 2006. 6
22. K. Kanatani and Y. Sugaya High Accuracy Fundamental Matrix Computation and Its Performance Evaluation. *IEICE - Trans. Inf. Syst.*, 579-585, 2007. 3

Image-Based Texturing of Dynamic 3-D Models

Zsolt Jankó¹

janko@imagine.enpc.fr

Jean-Philippe Pons²

jean-philippe.pons@cstb.fr

¹ IMAGINE Université Paris-Est and INRIA Rhône-Alpes / LJK, Grenoble, France

² IMAGINE Université Paris-Est, CSTB, Sophia-Antipolis, France

Abstract

In this paper we present a method for texturing dynamic 3-D models using calibrated video sequences from multiple viewpoints. To create the texture atlas, we fully exploit the very high redundancy in the input video sequences, by adopting an actual spatio-temporal perspective, instead of independent frame-by-frame computations. The main contribution is twofold. First, the amount of texture data is drastically reduced by eliminating redundancy, which greatly accelerates rendering and helps portability. Second, using several different viewpoint/time appearances of the scene, we can recover from low resolution, grazing views, and reduce the annoying effects of high-lights, shadows and occlusion. Altogether, our method allows the synthesis of novel views from a small quantity of texture data, with an optimal visual quality throughout the sequence, with minimally visible color discontinuities, and without flickering artifacts. These properties are demonstrated on real datasets.

1. Introduction

1.0.0.1. Motivation. In the recent years, several effective methods for the automatic generation of spatio-temporal models of dynamic scenes from video have been proposed [1, 3, 7, 8, 10, 11, 15, 17, 18, 19, 21]. However, capturing, processing and displaying the visual attributes, such as color, of such models, has been quite overlooked so far.

Surely, the case of static scenes, *i.e.* 3-D modeling from photographs, has been extensively studied: there exist several established techniques [4, 13, 14, 20] for creating image-based texture atlases, while avoiding visual artifacts such as color discontinuities, ghosting or blurring, which typically arise from photometric and geometric inaccuracies (varying light conditions and camera responses, non-Lambertian reflectance, imperfect camera calibration, approximate shape, *etc.*).

But the case of spatio-temporal models presents additional challenges. As the reflectance properties of the scene typically remain constant through time, the time redundancy in the input video sequences is very high. Fully exploiting this redundancy requires to adopt an actual spatio-temporal perspective, beyond independent frame-by-frame texture atlases. While the latter involve a prohibitively huge amount of texture data, *spatio-temporal texture atlases* are expected to be much more concise. Potentially, they could also take

advantage of time redundancy to recover from ambiguities and deficiencies in the input data.

1.0.0.2. Previous work. To our knowledge, there are only a few notable works on spatio-temporal texture atlases. In [17] the authors propose to warp all images in a common planar parameterization of the animated mesh, in order to recover spatially varying surface reflectance properties. However, this parameterization-based approach involves image resampling and loss of visual detail, and requires to carefully position cuts on the surface to avoid unacceptable distortion. In contrast, in the vein of several successful static 3-D methods [4, 13, 14, 20], we take advantage of the projective transformations from the moving surface to the input video frames, which constitute natural and optimal mappings.

In [2, 23] average textures are computed. First, average images are generated considering all the cameras, but separately for each frame. Then, these average images are merged into one final texture, by substituting the texture of invisible patches by the texture from the closest frame where it is visible [2], or by averaging over frames [23]. Due to blending, the results of these methods suffer from undesirable artifacts, such as ghosting or blurring, that we avoid.

The general principle of our method is to compute an optimal partition of the spatio-temporal surface of the scene, into patches associated to the different input images. The

problem is cast as a *Markov random field* optimization: to this extent, our work is most closely related to those of ^{4, 13} for static scenes. In this paper, we reformulate and extend these works, in order to meet the specific requirements and issues of spatio-temporal scenes.

1.0.0.3. Contribution. Our method enjoys several remarkable features. First, we drastically cut down on the amount of texture data, and thereby we greatly enhance the portability and the rendering efficiency of the model. Second, we gather the numerous different viewpoint/time appearances of the scene, so as to recover from low resolution, grazing views, highlights, shadows and occlusions which affect some regions of the spatio-temporal model[†]. Altogether, our method allows the synthesis of novel views from a small quantity of texture data, with an optimal visual quality throughout the sequence, with minimally visible color discontinuities, and without flickering artifacts. These properties are demonstrated on real datasets.

2. Problem Formulation and Solution

In this paper, we focus on a representation of dynamic scenes as *animated meshes*. An animated mesh consists in a sequence of meshes with a fixed connectivity (rather than unrelated meshes), whose time-varying vertex positions sample the trajectories of material points. It is a widely used representation in computer graphics, especially in computer animation.

This choice is not restrictive since there exist several methods for producing animated meshes of real dynamic scenes, either directly from video ^{3, 7, 8, 10, 15, 17, 21}, or from time-varying point clouds ^{16, 22} (the latter being obtained from video or from fast 3-D scanning hardware).

In the following, we consider a dynamic scene, imaged by N calibrated and synchronized video sequences composed of T frames, and approximated by an animated mesh with F polygonal faces. We note:

- $I_{n,t}$, $n \in \{1..N\}$, $t \in \{1..T\}$ the input images,
- $f_{k,t}$, $k \in \{1..F\}$, $t \in \{1..T\}$ the faces of the animated mesh at the different time instants.

2.1. Principle

Our method is based on two central assumptions. The first assumption is that the *reflectance properties* of the surface do not change through time. Please note that this does not exclude *appearance* changes between images, caused by non-Lambertian reflectance, varying shading, shadows and highlights along scene motion, or varying lighting conditions.

[†] One should note that highlights and shadows are necessary for photo-realistic rendering, but they have to be eliminated from the input videos and then synthesized depending on the new environment.

Once this clarification is made, it appears that an overwhelming majority of real-world scenes fulfill this assumption. The second assumption is that a mesh face corresponds to a same material patch throughout the sequence. The animated mesh representation precisely enforces this property.

Our method exploits these two assumptions to estimate a *normalized appearance* of the surface. Let us consider a face of the animated mesh, and the set of input images in which it is visible. Our rationale is that among these numerous different viewpoint/time appearances of the face, one or several of them are likely to approach ambient lighting conditions, and in particular to be exempt from shadow and highlights. By assigning each face of the animated mesh to one of these adequate input images, we can assemble a spatio-temporal texture atlas which allows the synthesis of *normalized* views of the dynamic scene, from any viewpoint and at any time instant.

Using a constant texture source for a face throughout the sequence has many desirable outcomes. First, we drastically cut down on the amount of texture data. The latter do not scale with the number of frames anymore. Second, if a region of the dynamic scene is visible once (in any time frame, from any input viewpoint), it can be rendered throughout the sequence. If a region is out-of-shadow once, we can discard the shadow throughout the sequence. The same benefits apply to highlights, texture resolution, and so on. Finally, we completely eliminate *flickering artifacts*, *i.e.* small color fluctuations in the animation, to which human eyes are very sensitive.

At this point, a clarification is needed. Rendering regions of the scene that are not visible in any input image at this time seems questionable at first sight: no information about the very geometry of these regions can be expected from the input data. However, most spatio-temporal reconstruction techniques are still able to infer relevant geometry and motion there, by assuming the spatial and temporal coherence of the scene. In turn, our method is able to infer texture in these regions.

The assignment of faces of the animated mesh to input images can be encoded by a labeling function

$$\mathcal{L} : \{1..F\} \rightarrow \{1..N\} \times \{1..T\}, \quad (1)$$

such that $\forall t$, face $f_{k,t}$ is textured from image $I_{\mathcal{L}(k)}$. To be more exact, faces that are not visible in any input image (in any time frame, from any input viewpoint) are discarded altogether, and are not considered in the above equation and in further discussions. Also, we may want to consider only a representative subset of the original time frames, in order to keep the computational complexity of our method sustainable for very long sequences. The rationale behind the latter simplification is that, with a sufficient number of frames with enough variety, the spatio-temporal texture atlas is very close to optimal, and is not significantly further improved by

supplemental frames. Remarkably, such an atlas can still be used to synthesize novel views of all original time frames.

The labeling (1) induces a partition of the animated mesh into *patches*. On the one hand, all faces inside a patch get their texture from the same image, so color is continuous across edges interior to the patch. On the other hand, at patch boundaries, *i.e.* at edges between faces with different labels, photometric and geometric inaccuracies in the data, in particular approximate geometry and motion, imperfect camera calibration, are likely to cause visually annoying color discontinuities (*seams*). Our method is able to compensate for most of these perturbations. It computes a labeling which optimizes visual quality, in some sense formally defined below, while minimizing the visibility of seams.

2.2. Variational Formulation

More specifically, we adopt a variational formulation: the optimality of a labeling is quantified by an energy functional composed of two terms, measuring the local visual quality and the visibility of seams, respectively:

$$E(\mathcal{L}) = E_{\text{quality}}(\mathcal{L}) + \mu E_{\text{seams}}(\mathcal{L}), \quad (2)$$

where μ denotes a weighting factor. In the rest of this subsection, we detail some possible definitions of these two energy terms. Subsection 2.3 describes the minimization procedure applied to the energy functional.

The visual quality term E_{quality} is local: it does not consider interactions between neighboring regions of the model. Hence we write it as a sum over faces:

$$E_{\text{quality}}(\mathcal{L}) = \sum_{k=1}^F \Phi_k(\mathcal{L}(k)), \quad (3)$$

where $\Phi_k(n, t)$ quantifies how appropriate image $I_{n,t}$ is for texturing faces $f_{k,\cdot}$ of the animated mesh. We can use different criteria to assess this quality.

In previous work on static scenes, several strategies have been used. In ¹³, the angle between viewing direction and face normal is proposed. In ⁴, the area of the projected face is advocated instead; in our context, this would write

$$\Phi_k(n, t) = -\text{area}[\Pi_n(f_{n,t})], \quad (4)$$

where Π_n denotes the projection from 3-D space to video sequence n . The latter choice would lead to an easily interpretable definition of visual quality: the total number of *texels* (texture elements) on the animated mesh.

Nevertheless, the above definitions mistakenly identify visual quality with visual detail. They fail to account for photometric aspects such as shadows and highlights. If the 3-D positions of light sources are known, we can easily estimate shadow and highlight regions on the spatio-temporal surface, and define visual quality as the total number of *shadow-free and highlight-free* texels on the animated mesh.

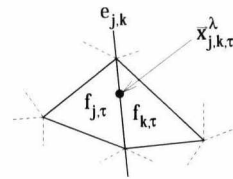


Figure 1: Notations of adjacent faces for term E_{seams} .

As this additional information is not available in practice for all datasets, we propose an alternative approach. Given a face, we compute its average projected area over all images where it is visible. To guarantee sufficient visual detail, we discard all images below average, *i.e.* we set Φ_k to infinity for these images. For the remaining possible images, we set Φ_k to the difference between the actual face intensity and the median of intensities. Here again, we compute the median over all images where the face is visible. Unfortunately, we could not test this criterion in this paper, for lack of datasets with significant shadows and highlights.

The second term E_{seams} measures how smoothly the color changes passing from one face to an adjacent. If the two faces take their textures from the same image, *i.e.* their label is the same, then the borderline between them is continuous. On the other hand, neighboring faces with different labels are likely to cause seams at the common edge. In order to minimize seam visibility, the second energy term is defined as the integral along the seams of color discrepancy between bordering images.

Further notations are needed to write a formal expression of this term. (See Figure 1.) Let us denote by $e_{j,k}$ a non-border edge of the animated mesh, adjacent to faces $f_{j,\cdot}$ and $f_{k,\cdot}$. We note $\bar{x}_{j,k,\tau}^\lambda, \lambda \in [0, 1]$ a linear parameterization of $e_{j,k}$ at time τ . Notice that the color at a point $\bar{x}_{j,k,\tau}^\lambda$ on an edge does not depend on the time frame τ , but only on the selected input image (n, t) . We note $c_{j,k}^\lambda(n, t)$ the color at $\bar{x}_{j,k,\tau}^\lambda$, extracted from image $I_{n,t}$:

$$c_{j,k}^\lambda(n, t) = I_{n,t} \circ \Pi_n(\bar{x}_{j,k,\tau}^\lambda). \quad (5)$$

With these notations in hand, E_{seams} writes as a sum over the set \mathcal{E} of all non-border edges:

$$E_{\text{seams}}(\mathcal{L}) = \sum_{e_{j,k} \in \mathcal{E}} \|e_{j,k}\| \Psi_{j,k}(\mathcal{L}(j), \mathcal{L}(k)), \quad (6)$$

$$\Psi_{j,k}(\ell, \ell') = \int_0^1 \|c_{j,k}^\lambda(\ell) - c_{j,k}^\lambda(\ell')\| d\lambda. \quad (7)$$

$\|e_{j,k}\|$ is computed as the average length of the edge over all frames.

2.3. Optimization Procedure

It must be noted that $\Psi_{j,k}$ is a semi-metric: $\forall \ell, \ell', \ell'' \in \{1..N\} \times \{1..T\}$,

- $\Psi_{j,k}(\ell, \ell) = 0$,
- $\Psi_{j,k}(\ell, \ell') = \Psi_{j,k}(\ell', \ell) \geq 0$,
- $\Psi_{j,k}(\ell, \ell'') \leq \Psi_{j,k}(\ell, \ell') + \Psi_{j,k}(\ell', \ell'')$.

This still holds with any metric on colors instead of the usual Euclidean distance in RGB color space.

This has an important practical consequence: it allows us to minimize the energy functional (2) with α -expansion^{6,12}. It consists in translating the labeling problem, which is generally NP-hard, to a succession of binary minimum cut problems. Efficient solutions to these min-cut problems are described in⁵. The whole process monotonically decreases the energy and is guaranteed to converge to a *strong local minimum*, thereby ensuring a close-to-optimal seam placement.

In our implementation, we use the graph cuts minimization software by O. Veksler (<http://www.csd.uwo.ca/~olga/code.html>) and by V. Kolmogorov (<http://www.adastral.ucl.ac.uk/~vladkolm/software.html>).

2.4. Texture Atlas Creation

We could synthesize novel views from the input video sequences using multiple passes of projective texture mapping⁹. However, the creation of a single rectangular texture map is very desirable: it increases the rendering efficiency and allows to output portable animated 3D formats. To build such a spatio-temporal texture atlas, we consider the binary masks representing the useful regions in the different images: $M_{n,t}$ is the projection in $I_{n,t}$ of the associated patch:

$$M_{n,t} = \Pi_n \left[\bigcup_{\mathcal{L}(k)=(n,t)} f_{k,t} \right]. \quad (8)$$

We first apply a morphological dilation to the masks with a square structural element of a few pixels, in order to provision for automatic texture minifying during rendering. We then compute a connected component decomposition, yielding a list of texture fragments. We pack the latter using a classical first-fit decreasing strategy: we place the fragments in decreasing order of size, at the first available slot found along a scanline search in the atlas. Finally, we set the texture coordinates of the vertices of the animated mesh accordingly. Thus, the final output of our algorithm is compatible with standard 3D viewers.

3. Experimental Validation

3.1. Input Datasets

In order to validate our method under real conditions, we tested it on three challenging datasets: 'crane', 'samba'

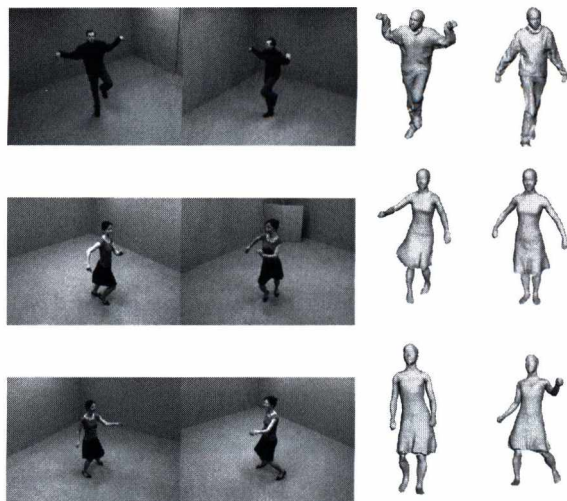


Figure 2: Some views of our test datasets. From left to right: two input images, two views of the animated mesh. From top to bottom: 'crane', 'samba' and 'swing' datasets.

and 'swing', that were made available online by the authors of²¹ at http://people.csail.mit.edu/drdaniel/mesh_animation/.

Each dataset consists of 8 calibrated and synchronized 1600×1200 color video streams. 'crane' and 'samba' are composed of 175 frames. 'swing' is composed of 150 frames. The datasets also include an animated mesh of the dynamic scene, of approximately 10K vertices and 20K faces, obtained in accordance with²¹. Figure 2 displays some views of these datasets.

3.2. Compared Methods

On each of the datasets, we compare four different methods, which we call 'single-frame', 'single-frame optimized', 'multi-frames' and 'multi-frames optimized'.

The 'single-frame' and 'single-frame optimized' methods both compute independent frame-by-frame texture atlases. The 'single-frame' method greedily maps each face to the highest-quality input image of the current frame. The 'single-frame optimized' achieves a trade-off between visual quality and visibility of seams, similarly to^{4,13}.

The 'multi-frames' and 'multi-frames optimized' methods both compute a single spatio-temporal texture atlas. To limit computational expense, we build this atlas from a subset of the input time frames, namely 10 uniformly distributed time frames, with no noticeable deterioration of the results. The 'multi-frames' method greedily maps each face to the highest-quality input image among all viewpoints and all se-

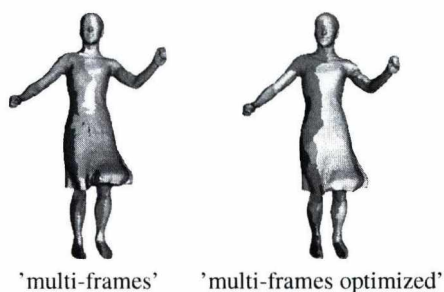


Figure 6: Color-coded partition of the surface obtained on the 'samba' dataset, with two different methods.



Figure 7: A sample spatio-temporal texture atlas.

lected time frames:

$$\mathcal{L}(k) = \arg \max_{n,t} \phi_k(n,t).$$

Finally, the 'multi-frames optimized' method is the one described in this paper. Let us mention that the 'multi-frames optimized' method degenerates to the 'multi-frames' method when setting the weighting factor μ of E_{seams} to zero.

In all these experiments, we use the number of texels (4) as the visual quality measure.

3.3. Results

The results of the four aforementioned methods on the 'crane', 'samba' and 'swing' datasets are compared in Figures 3, 4 and 5, respectively. Due to space limitations, we only show three views for each method and each dataset, synthesized at three different time instants.

Also, for illustration purposes, Figure 6 displays the partition of the surface obtained on the 'samba' dataset with the 'multi-frames' and the 'multi-frames optimized' methods. Finally, Figure 7 displays a sample spatio-temporal texture atlas, obtained from the 'swing' dataset.

3.4. Discussion

These results clearly demonstrate one major advantage of multi-frames methods over single-frame methods: the ability to cover temporarily hidden regions as well, which results in significantly smaller untextured regions in novel views.

Another drawback of single-frame methods is also apparent in these results, particularly in the accompanying videos: the incoherence of surface partition across time frames produces small color fluctuations, known as flickering artifacts, to which human eyes are very sensitive.

The poor results of the 'multi-frames' method reveal the important geometric inaccuracy of the input datasets (approximate calibration, geometry and motion). This inaccuracy prevents from naively assembling a satisfactory spatio-temporal texture atlas from input images that are very distant, in viewpoint or in time. Remarkably, our method ('multi-frames optimized') turns out to be robust to such imperfect data, indicating that a global optimization of surface partition is essential for the success of our approach.

At the same time, our experiments highlight a limit of spatio-temporal texture atlases: if very detailed geometry and motion, such as creases of clothes and facial expression, are not modeled in the animated mesh, they cause a violation of our central 'constant reflectance' assumption. As a result, these variations are averaged out by our approach. This explains the visually intriguing static faces visible in our results.

Besides, we should note that our method does not explicitly take photometry into account, and although our method could evolve in this direction, in this paper we deliberately adopt an orthogonal approach: we estimate a plausible diffuse map of the scene by automatically discarding complex photometric effects. Our method, though it has limitations, is simple and easy to implement, and our results show significant advantages compared to frame-by-frame texturing.

4. Conclusion

We have proposed a method to create high-quality spatio-temporal texture atlas for dynamic 3-D model. The texture map is built using a set of calibrated video sequences by exploiting its high redundancy. We represent the dynamic scene as animated mesh with fixed connectivity, and assume the reflectance properties of the surface to be constant through time, which is valid for most real-world scenes.

The main contribution is twofold. First, we drastically cut down on the amount of texture data, and thereby we greatly enhance the portability and the rendering efficiency of the model. Second, we gather the numerous different viewpoint/time appearances of the scene, so as to recover from low resolution, grazing views, highlights, shadows and occlusions which affect some regions of the spatio-temporal model. We have demonstrated the advantages of our method compared to single-frame texturing on real datasets.

Acknowledgment. This work was supported by the French National Agency for Research (ANR) under grant ANR-06-MDCA-007.

References

1. E. Aganj, J.-P. Pons, F. Ségonne, and R. Keriven. Spatio-temporal shape from silhouette using four-dimensional Delaunay meshing. In *Proc. ICCV'07*, 2007.

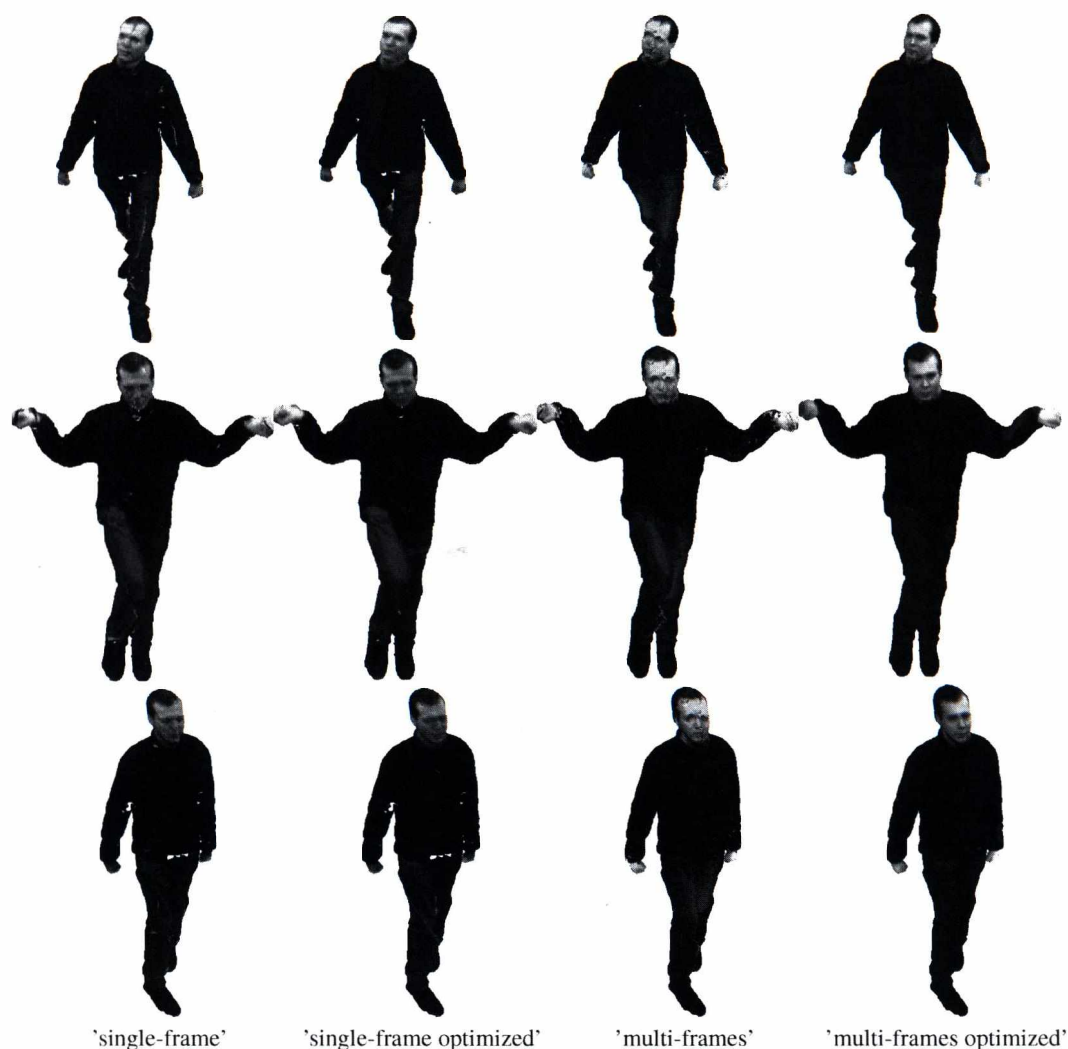


Figure 3: Results on the 'crane' dataset. See text for more details.

2. N. Ahmed, E. de Aguiar, C. Theobalt, M. Magnor, and H.-P. Seidel. Automatic generation of personalized human avatars from multi-view video. In *Proc. VRST'05*, pages 257–260, 2005.
3. N. Ahmed, C. Theobalt, C. Rössl, S. Thrun, and H.-P. Seidel. Dense correspondence finding for parametrization-free animation reconstruction from video. In *Proc. CVPR'08*, 2008.
4. C. Allène, J.-P. Pons, and R. Keriven. Seamless image-based texture atlases using multi-band blending. In *Proc. ICPR'08*, 2008.
5. Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, 2004.
6. Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.
7. E. de Aguiar, C. Stoll, C. Theobalt, N. Ahmed, H.-P. Seidel, and S. Thrun. Performance capture from sparse multi-view video. In *Proc. ACM SIGGRAPH*, 2008.
8. E. de Aguiar, C. Theobalt, C. Stoll, and H.-P. Seidel. Marker-less deformable mesh tracking for human shape and motion capture. In *Proc. CVPR'07*, 2007.
9. P. E. Debevec, C. J. Taylor, and J. Malik. Modeling and rendering architecture from photographs: a hybrid geometry- and image-based approach. In *Proc. SIGGRAPH*, pages 11–20, 1996.



Figure 4: Results on the 'samba' dataset. See text for more details.

10. Y. Furukawa and J. Ponce. Dense 3D motion capture from synchronized video streams. In *Proc. CVPR'08*, 2008.
11. B. Goldlücke and M. Magnor. Space-time isosurface evolution for temporally coherent 3D reconstruction. In *Proc. CVPR'04*, volume 1, pages 350–355, 2004.
12. V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? In *Proc. ECCV'02*, pages 65–81, 2002.
13. V. S. Lempitsky and D. V. Ivanov. Seamless mosaicing of image-based texture maps. In *Proc. CVPR'07*, 2007.
14. C. Rocchini, P. Cignoni, C. Montani, and R. Scopigno. Acquiring, stitching and blending diffuse appearance attributes on 3D models. *The Visual Computer*, 18(3):186–204, 2002.
15. J. Starck and A. Hilton. Surface capture for performance based animation. *IEEE Computer Graphics and Applications*, 27(3):21–31, 2007.
16. J. Süßmuth, M. Winter, and G. Greiner. Reconstructing animated meshes from time-varying point clouds. *Computer Graphics Forum (Proc. SGP'08)*, 27(5):1469–1476, 2008.
17. C. Theobalt, N. Ahmed, H. Lensch, M. Magnor, and H.-P. Seidel. Seeing people in different light-joint shape, motion, and reflectance capture. *IEEE Transactions on Visualization and Computer Graphics*, 13(4):663–674, 2007.
18. K. Varanasi, A. Zaharescu, E. Boyer, and R. P. Horaud. Temporal surface tracking using mesh evolution. In *Proc. ECCV'08*, volume 2, pages 30–43, 2008.



Figure 5: Results on the 'swing' dataset. See text for more details.

19. S. Vedula, S. Baker, and T. Kanade. Image-based spatio-temporal modeling and view interpolation of dynamic events. *ACM Trans. on Graphics*, 24(2):240–261, 2005.
20. L. Velho and J. Sossai Jr. Projective texture atlas construction for 3D photography. *The Visual Computer*, 23(9):621–629, 2007.
21. D. Vlastic, I. Baran, W. Matusik, and J. Popović. Articulated mesh animation from multi-view silhouettes. *ACM Transactions on Graphics*, 27(3), 2008.
22. M. Wand, P. Jenke, Q. Huang, M. Bokeloh, L. Guibas, and A. Schilling. Reconstruction of deforming geometry from time-varying point clouds. In *Proc. SGP'07*, pages 49–58, 2007.
23. G. Ziegler, H. Lensch, M. Magnor, and H.-P. Seidel.

Multi-video compression in texture space. In *Proc. ICIP'04*, volume 4, pages 2467–2470, 2004.

Másodfokú közelítés implicit felületek síkbeli leképezésére

Molnár József, Csetverikov Dmitrij

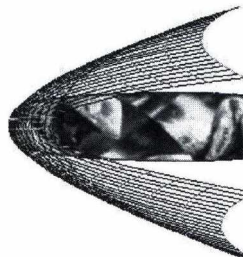
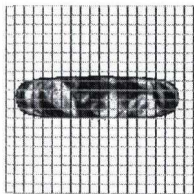
Magyar Tudományos Akadémia
Számítástechnikai és Automatizálási Kutatóintézet

Absztrakt

A Level-set módszerekben használt nem paraméteres, többnyire előjeles távolságfüggvénnyel adott felület-reprezentációval számos alkalmazásban találkozhatunk. Bizonyos alkalmazások az egymásnak megfeleltetett részletek elemzésével oldanak meg problémákat. Konkrétan a gépi látás olyan területein, ahol nagyfokú robusztusság követelmény, gyakran találkozunk a keresztkorreláció eszközeivel. Az egymásnak megfeleltetett képrészletek közötti korreláció pontos méréséhez több, esetenként egymásnak ellentmondó feltételnek kell megfelelni. Egyfelől a korrelációs ablak minél nagyobb mérete lenne kívánatos a lokálisan információszegény, vagy az összetéveszhető (ismétlődő) textúrákkal bíró felületek esetén. Másfelől viszont a méret növelése ellen hat a szokásos, felületek lokálisan síkokkal történő közelítése és/vagy a síkokra való vetületek képezésének elsőfokú közelítése. Célul tűzzük ki a fenti problémáktól mentes kvadratikus transzformáció levezetését.

Kulcsszavak: implicit felületek, vetületek közötti transzformáció, kvadratikus transzformáció

1 BEVEZETÉS



1. ábra: A probléma érzékeltetése

Gyakori eset, például a sztereó rekonstrukció problémánál, amikor fénykép- (video)felvételek elemzésekor kell megfeleltetéseket keresnünk a képrészletek között. A megfeleltetés egyértelműsége és a robusztusság követelménye gyakran vezet a keresztkorreláció használatához. Ez egy adott pont körüli képrészletek összehasonlításával történik. Figyelembe kell azonban venni a részletek között várható olyan eltéréseket (torzulások), amelyek a megfigyelt felület és a felvételeket készítő eszközök tulajdonságaiból előre kiszámíthatók. Ezek az előzetesen figyelembe vehető tulajdonságok az eszközök esetén vetítési függvények (kalibrációval), a megfigyelt felület esetén pedig annak valamilyen

invariáns differenciális mennyisége (tipikusan a normálvektora). Kérdés, hogy találhatunk-e olyan, gyakorlatban is használható, a szokásosnál magasabb rendű közelítéseket, amelyek használatával pontosabb rekonstrukcióra számíthatunk.

A fentieket formalizáljuk: a tér minden pontjának leképezését ismertnek tételezzük fel, azaz minden $\mathbf{S} \in R^3$ -hoz ismerjük az $x_i = x_i(\mathbf{S})$, $y_i = y_i(\mathbf{S})$ $i=1,2$ függvényeket. Meghatározandó az \mathbf{S} egy ismert pontjának az 1. kamerabeli (x_1, y_1) képétől (dx_1, dy_1) pixeles pozícióban levő pont közelítő helyzete a 2. kamerabeli (x_2, y_2) képéhez, azaz keressük a $dx_2 = \bar{f}(dx_1, dy_1)$, $dy_2 = \bar{g}(dx_1, dy_1)$ közelítő függvényeket, a (dx_1, dy_1) „átvetítését” (dx_2, dy_2) -be. A keresett függvények nyilván függenek a megfigyelt felületelem jellemzőitől és a teret a kamerákra képező függvényektől (1. ábra).

2 KÖZELÍTÉSEK ÉS TULAJDONSÁGAIK

Alább a gyakorlatban használt közelítések összefoglalása után bevezetjük a kvadratikus közelítést.

Nulladik fokú közelítés: a fronto-parallel modell

A modell feltételezése szerint (a végeredmény irányából indulva): $[dx_2, dy_2]^T = [dx_1, dy_1]^T$. Ebben a legegyszerűbb esetben azt feltételezzük a megfigyelt felületelem kicsiny környezete ugyanúgy látszik mindkét kamerából. Szigorú esetben ez

1. az azonos, egymáshoz képest a retinális síkban párhuzamosan eltoló kamerákkal (vagy axonometrikus vetítést, párhuzamosan eltoló kamerákkal), és
2. a retinális síkkal párhuzamos felületelemek megfigyelését

jelenti. A 2. feltétel miatt a kamerák retinális síkjának mindenképpen párhuzamos síkoknak kell lenniük, és ebben a síkban sem foroghatnak el egymáshoz képest. A gyakorlatban ezen feltételek egy része kis bázistávolságú (illetve a bázistávolsághoz képest távoli objektum) megfigyelést jelent kanonikus elrendezésű sztereó kamerapárral. Ezért az 1. feltétel egyszerűen teljesülhet, de a 2. feltétel teljesülésére általában nem számíthatunk.

Mivel a fronto-parallel modell a leképezési függvényekről, illetve a megfigyelt felület tulajdonságaiból információt nem tartalmaz, ezért joggal nevezhetjük nulladfokú közelítésnek. Gyakorlati alkalmazása ma már ritka.

Kollineáció és affin transzformáció

Az egyik leggyakrabban közelítő modell. A vetítési függvényekről feltételezi, hogy a lyukkamera modellnek felelnek meg (ez a feltételezés széles körben használt), a megfigyelt felületelemet pedig síknak tekinti. A levezetést mellőzve a képrészletek közötti transzformációra a következő kifejezés adódik:

$$dx_2(dx_1, dy_1, x_2) = \frac{x_2 + \frac{h_{1x}dx_1 + h_{1y}dy_1}{h_{3x}x_1 + h_{3y}y_1 + h_{3z}}}{1 + \frac{h_{3x}dx_1 + h_{3y}dy_1}{h_{3x}x_1 + h_{3y}y_1 + h_{3z}}} - x_2$$

$$dy_2(dx_1, dy_1, y_2) = \frac{y_2 + \frac{h_{2x}dx_1 + h_{2y}dy_1}{h_{3x}x_1 + h_{3y}y_1 + h_{3z}}}{1 + \frac{h_{3x}dx_1 + h_{3y}dy_1}{h_{3x}x_1 + h_{3y}y_1 + h_{3z}}} - y_2$$

A h_{iz} az ún. homográfia mátrixának elemei, amely tartalmazza a sík normál-egységvektorát és a kamerák egymáshoz képesti (tetszőleges, merevtest-szerű) elmozgatásának adatait. A fenti egyenletek sajnos nemcsak a $[dx_1, dy_1]^T$ vektort tartalmazzák a jobb oldalon, hanem az $[dx_2, dy_2]^T$ pontokat is, azaz nem mindegy, hogy a megfigyelt részlet 2. kamera képén hova esik. Szokás még a kollineáció linearizálásával nyert, ún. affin transz-

formáció használata is. Gyakorlati alkalmazásuk széles körben elterjedt a kamera kalibrációtól [1] a sztereó rekonstrukcióig [2].

Elsőfokú közelítés: a lineáris transzformáció

A későbbiek jobb megértése végett most egy más megközelítéssel (egyfajta elemi úton) vezetünk le lineáris transzformációt a képrészletek között. Ez az elemi út jól mutatja, hogy itt pontosan milyen jellegű közelítéseket alkalmazunk:

1. A felületelemet paraméteres formában (általános paraméterezéssel) megadhatónak tételezzük fel: $\mathbf{S} = x(u, v)\mathbf{e}_x + y(u, v)\mathbf{e}_y + z(u, v)\mathbf{e}_z$

2. Ismerjük a kamerák leképező függvényeit, az $x_1 = x_1(x, y, z)$, $y_1 = y_1(x, y, z)$ és $x_2 = x_2(x, y, z)$, $y_2 = y_2(x, y, z)$ transzformációkat.

3. Ekkor a láncszabály szerint a felület egy pontjának képe körüli elmozdulás közelítése – miközben a felületen egy $[du, dv]$ -vel definiált pontba tovább lépünk – csak a lineáris tagokat megtartva, mátrixos alakban így írható fel:

$$\begin{bmatrix} dx_1 \\ dy_1 \end{bmatrix} = \begin{bmatrix} \frac{\partial x_1}{\partial x} \frac{\partial x}{\partial u} + \frac{\partial x_1}{\partial y} \frac{\partial y}{\partial u} + \frac{\partial x_1}{\partial z} \frac{\partial z}{\partial u} & \frac{\partial x_1}{\partial x} \frac{\partial x}{\partial v} + \frac{\partial x_1}{\partial y} \frac{\partial y}{\partial v} + \frac{\partial x_1}{\partial z} \frac{\partial z}{\partial v} \\ \frac{\partial y_1}{\partial x} \frac{\partial x}{\partial u} + \frac{\partial y_1}{\partial y} \frac{\partial y}{\partial u} + \frac{\partial y_1}{\partial z} \frac{\partial z}{\partial u} & \frac{\partial y_1}{\partial x} \frac{\partial x}{\partial v} + \frac{\partial y_1}{\partial y} \frac{\partial y}{\partial v} + \frac{\partial y_1}{\partial z} \frac{\partial z}{\partial v} \end{bmatrix} \begin{bmatrix} du \\ dv \end{bmatrix}$$

Vagy $d\mathbf{r}_1 = \mathbf{A}_1 d\mathbf{u}$ (Az \mathbf{A}_1 a leképezés Jacobi mátrixa)

4. A 2. kamerára ugyanez a művelet elvégezve, valamint a $[du, dv]$ vektort az 1. kamera egyenletéből kifejezve (\mathbf{A}_1^{-1} szorzással), és a 2. kamera egyenletébe helyettesítve megfelelő átrendezések után kapjuk a következő formulát:

$$\begin{bmatrix} dx_2 \\ dy_2 \end{bmatrix} = \mathbf{A}_2 \mathbf{A}_1^{-1} \begin{bmatrix} dx_1 \\ dy_1 \end{bmatrix} = \mathbf{A} \begin{bmatrix} dx_1 \\ dy_1 \end{bmatrix}, \text{ ahol az}$$

\mathbf{A} mátrix:

$$5. \mathbf{A} = \mathbf{A}_2 \mathbf{A}_1^{-1} = \frac{1}{|\nabla x_1 \mathbf{n} \nabla y_1|} \begin{bmatrix} |\nabla x_2 \mathbf{n} \nabla y_1| & |\nabla x_1 \mathbf{n} \nabla x_2| \\ |\nabla y_2 \mathbf{n} \nabla y_1| & |\nabla x_1 \mathbf{n} \nabla y_2| \end{bmatrix}$$

jelölésben: $|\mathbf{uvw}| = \mathbf{u} \cdot (\mathbf{v} \times \mathbf{w})$

amely egyenletből látszik, hogy a lineáris transzformáció mátrixának elemei a projekciók gradienseinek, és a felület normál-egységvektorainak vegyes szorzatából álló mennyiségek. Azaz mind a vetítési függvényeket, mind a megfigyelt elemi felületet lineárisan közelítő függvényekből áll össze. Ha az \mathbf{A} mátrixot történetesen egy olyan funkcionálban alkalmazzuk, ahol az ismeretlen függvényünk éppen a megfigyelt felület, akkor az ismeretlen függvény aspektusából felírható:

$$\mathbf{A} = \mathbf{A}(x, y, z, n_x, n_y, n_z) = \mathbf{A}(\mathbf{S}, \mathbf{n}).$$

Kvadratikus transzformáció

A fenti lineáris transzformáció bizonyos esetekben elégtelen lehet:

1. Különböző okok miatt, pl. nagyobb homogén régiók, ismétlődő mintázat esetén nagyobb felületelemek összehasonlítására lenne kívánatos. Ekkor a megfigyelt felületelem pontjainak az eltávolodása az érintősíktól (azaz a felület görbülete) nem elhanyagolható a vizsgált tartományban.
2. Viszonylagosan nagy bázistávolság esetén a leképezés lineáris közelítése nem kielégítő.

A kollineáció a vetítési függvényeket (csak lyukkamera modell esetén) egzakt formában tartalmazza, de kizárólag akkor használható, ha megfigyelt felületelem sík. Ha az 1. probléma is fennáll, akkor erről a lehetőségről le kell mondanunk. Az elsőfokú közelítés mintájára megkísérelhető olyan másodfokú közelítő egyenletek levezetése, ahol mind a projekciós függvények, mind a megfigyelt felületelem kvadratikus közelítettek.

A kvadratikus transzformáció levezetésének elvi lépései

A kvadratikus transzformáció létesítése a képrészletek között nem olyan triviális feladat, mint a lineáris transzformációnál volt. Egyrészt a fellépő differenciálok a második hatványon is szerepelnek, azaz nem folyamodhatunk egyszerű inverzképzéshez, másrészt a paraméterezéstől való elszakadás, azaz az invariáns mennyiségek meghatározása nem megy egyszerű eszközökkel (átrendezések, átzárójelezések). Az alábbiakban a kvadratikus transzformáció levezetésének elvi lépéseit foglaljuk össze.

1. lépés

A felületet most is általános koordinátákkal adott:

$$\mathbf{S} = x(u, v)\mathbf{e}_x + y(u, v)\mathbf{e}_y + z(u, v)\mathbf{e}_z \quad (1)$$

Ennek a felületnek egy elemi részletét két kamerával (i. és j.) figyeljük meg. A megfigyelt részlet egy pontjának képp koordinátáit ismerjük mindkét kamerán, ezek: $[x_i, y_i]^T$ és $[x_j, y_j]^T$ koordináták.

A felület egy rögzített pontja körüli – ennek a pontnak a pixeles koordinátái $[x_i, y_i]^T - (du, dv)$ elmozdulás hatása az i. kameraképen, amennyiben a másodfokú Taylor polinomos közelítésről feltesszük, hogy kielégítő:

$$\begin{aligned} dx_i &= \frac{\partial x_i}{\partial u} du + \frac{\partial x_i}{\partial v} dv \\ &+ \frac{1}{2} \left(\frac{\partial^2 x_i}{\partial u^2} du^2 + 2 \frac{\partial^2 x_i}{\partial u \partial v} dudv + \frac{\partial^2 x_i}{\partial v^2} dv^2 \right) \\ &= Q_{xi}^{jel}(du, dv) \\ dy_i &= \frac{\partial y_i}{\partial u} du + \frac{\partial y_i}{\partial v} dv \\ &+ \frac{1}{2} \left(\frac{\partial^2 y_i}{\partial u^2} du^2 + 2 \frac{\partial^2 y_i}{\partial u \partial v} dudv + \frac{\partial^2 y_i}{\partial v^2} dv^2 \right) \quad (2) \\ &= Q_{yi}^{jel}(du, dv) \end{aligned}$$

Most a lineáris transzformáció levezetésének mintájára, a 4. pontjának megfelelően a (2) másodfokú egyenletrendszer kellene (du, dv) -re megoldanunk.

2. lépés

Ehelyett azonban (és ez a legjelentősebb eltérés), az inverz transzformációról feltesszük, hogy szintén jól közelíthető egy másodfokú Taylor sorral:

$$\begin{aligned} du &= Q_u(dx_i, dy_i) \\ dv &= Q_v(dx_i, dy_i) \end{aligned} \quad (3)$$

ahol azonban a sorfejtés együtthatóit ismeretlennek tételezzük fel. A (3) egyenleteket (2)-be helyettesítve, továbbá feltéve, hogy a dx_i, dy_i differenciálok kicsinyek ugyan, de tetszőlegesen, végül a másodfokú magasabb fokú tagokat elhagyva (azaz ahol egy szorzat tényezői hatványkitevőinek összege kettőnél nagyobb), a (3) ismeretlen együtthatóira független két ismeretlenes lineáris egyenletrendszereket kapunk (összesen 5 darabot), amelyek könnyen megoldhatók. Megközelítésünk helyességét igazolja, hogy a (3) elsőfokú együtthatókra a (2) elsőfokú tagjaiból álló mátrix (a Jacobi mátrix) inverze adódik csakúgy mint a lineáris transzformáció esetében, tehát (3) kvadratikus egyenletek a lineáris tagjai a éppen az előző pontban leírt lineáris transzformációt valószínűsítik meg, a további tagok a tiszta másodrendű hatásokért felelősek.

3. lépés

Most az előző lépésben kapott (du, dv) értékeket egy, a második kamerára felírt másodfokú Taylor polinomba helyettesítjük (a lineáris transzformáció 4. lépésével analóg módon), és a kapott egyenleteket az előző lépéshez hasonlóan egyszerűsítjük, azaz minden olyan tagot elhagyunk, ahol a tényező (du, dv) szerinti össz-hatványa kettőnél nagyobb. Ezzel a lépéssel a kívánt formát kapjuk:

$$\begin{aligned} dx_j &= Q_{xj}(dx_i, dy_i, du, dv) \\ dy_j &= Q_{yj}(dx_i, dy_i, du, dv) \end{aligned} \quad (4)$$

Az egyetlen probléma, hogy az egyenletek a kiinduló lépésnek megfelelően a felületre jellemző mennyiségeket az (ismeretlen paraméterezésnek megfelelő) paraméteres formában tartalmazzák. Célunk tehát a (4)-ben du és dv invariáns mennyiségekkel helyettesítése.

4. lépés: invariáns egyenletek

Az utolsó lépés a paraméteres mennyiségek kiküszöbölése, hiszen implicit felületek esetén a paraméterezés ismeretlen. A lépés most egyáltalán nem olyan triviális mint a lineáris transzformáció esetén, ahol szinte csak „észre kellett venni” a paraméteres mennyiségek mögötti invariáns tartalmat.

A fellépő mennyiségek két csoportba sorolhatók. Az első csoportba tartozók a felületnek csak az elsőrendű deriváltjait tartalmazzák, és egyszerű átrendezéssel, átzárójelezéssel felismerhető a felületre jellemző elsőrendű invariáns mennyiség: a felület normál-egységvektora. Ezek a mennyiségek (tenzorok) felelősek egyébként a perspektivikus hatásért.

A második csoport azonban nem ilyen egyszerű. A paraméteres forma tartalmazza a felület másodrendű paraméteres mennyiségeit is, és semmilyen azonos átalakítással nem sikerült invariáns formára hozni. A cél elérése érdekében szükség volt egy különleges konstrukcióra, amely a felület érintősíkjaiban speciális paraméterezéseket (lokális bázisokat) valósít meg, és amelyek segítségével a felületelem másodrendű mennyiségei invariánsokat tartalmazó alakot öltenek! Ezt a paraméterezést az egyszerűség kedvéért egy megfelelően választott konstans vektor generálja.

A paramétervonalakat a következőképpen konstruáljuk: a választott konstans vektor meghatároz egy párhuzamos síkhalmazt. Ezeknek a felülettel való metszetét nevezzük „u” vonalnak, és az erre merőleges, a felületen futó vonalat pedig „v” vonalnak. Ha ezen paramétervonalakat ívhosszuk szerint járjuk be, akkor érintővektorok és a felületi normálvektor ortonormált bázisrendszeret alkotnak.

A konstrukcióból adódóan a levezetett transzformáció a felület azon pontjain nem értelmezett, ahol a választott sík egybeesik a felület érintősíkjával. A szokásos eljárás ennek a kiküszöbölésére, hogy mindig a „világ”- koordinárendszer biztosította síkok közül a legmegfelelőbbet választjuk.

Végül ezekben a bázisokban kifejezve a (4) egyenletben szereplő, a felületre jellemző másodrendű mennyiségek invariáns alakot vesznek fel. A fentieket formalizálva ($\mathbf{S}_u, \mathbf{S}_v, \mathbf{n}$ a lokális rendszer):

$$s = |\mathbf{n} \times \mathbf{c}|, \mathbf{S}_u = \frac{1}{s} \mathbf{n} \times \mathbf{c}, \mathbf{S}_v = \mathbf{n} \times \mathbf{S}_u, \quad (5)$$

$$(\mathbf{N} = \mathbf{S}_u \times \mathbf{S}_v = \mathbf{n})$$

Ekkor: $\partial_u \mathbf{S}_u = (\mathbf{S}_u \nabla) \cdot \mathbf{S}_u$, $\partial_u \mathbf{S}_v = (\mathbf{S}_v \nabla) \cdot \mathbf{S}_u$,
 $\partial_v \mathbf{S}_u = (\mathbf{S}_u \nabla) \cdot \mathbf{S}_v$, $\partial_v \mathbf{S}_v = (\mathbf{S}_v \nabla) \cdot \mathbf{S}_v$, és ahol $\mathbf{c} \in (\mathbf{e}_x, \mathbf{e}_y, \mathbf{e}_z)$, \mathbf{c} nem párhuzamos \mathbf{n} -nel!

A négyféle projekciós függvény gradiensei: $\nabla Z_K \in (\nabla x_K, \nabla y_K)$, $K = i, j$ által definiált tenzorok

az invariáns mennyiségekre átfogalmazott (4) egyenletek építőkövei:

$$\begin{aligned} \mathbf{T}_{\nabla Z_K} &= -(\nabla Z_K \cdot \partial_v \mathbf{S}_v) \mathbf{S}_u \mathbf{S}_u + (\nabla Z_K \cdot \partial_u \mathbf{S}_v) \mathbf{S}_u \mathbf{S}_v \\ &+ (\nabla Z_K \cdot \partial_v \mathbf{S}_u) \mathbf{S}_v \mathbf{S}_u - (\nabla Z_K \cdot \partial_u \mathbf{S}_u) \mathbf{S}_v \mathbf{S}_v \end{aligned} \quad (6)$$

$$\mathbf{U}_{\nabla Z_K} = [\mathbf{n}]_x \cdot (\nabla \nabla Z_K) \cdot [\mathbf{n}]_x \quad (7)$$

A pont a skaláris szorzás jele, az $[\mathbf{n}]_x$ a normál-egységvektor „kereszt tenzora”: $[\mathbf{n}]_x \cdot \mathbf{v} = \mathbf{n} \times \mathbf{v}$ minden \mathbf{v} -re. A \mathbf{T} tenzor fenti formája jól mutatja, hogy az érintősíkban értelmezett (kétdimenziós) tenzor, amely reprezentánsa az $\mathbf{S}_u, \mathbf{S}_v$ bázisban a zárójeles mennyiségekből álló mátrix. Amíg az \mathbf{U} tenzor kizárólag a vetítési függvény másodrendű jellemzőit tartalmazza (perspektíva), addig a \mathbf{T} -k a felület másodrendű jellegzetességeit (görbület) is. Ezek alapján a (4) a felületre jellemző mennyiségektől való függésében így írható fel:

$$\begin{aligned} dx_j &= Q_{xj}(\mathbf{S}, \mathbf{n}, \nabla \mathbf{n}) \\ dy_j &= Q_{yj}(\mathbf{S}, \mathbf{n}, \nabla \mathbf{n}) \end{aligned} \quad (8)$$

Azaz egy funkcionált minimalizáló ismeretlen \mathbf{S} függvény aspektusából a (8) egy olyan kvadratikus transzformáció, amely tartalmazza az ismeretlen felület mellett annak első- és másodrendű invariáns, differenciális mennyiségeit.

A kvadratikus transzformáció egyenletei

Jelölések:

A lineáris transzformáció mátrixának elemei:

$$\begin{aligned} a_{11} &= \frac{|\nabla x_j \mathbf{n} \nabla y_i|}{|\nabla x_i \mathbf{n} \nabla y_j|}, a_{12} = \frac{|\nabla x_i \mathbf{n} \nabla x_j|}{|\nabla x_i \mathbf{n} \nabla y_i|}, \\ a_{21} &= \frac{|\nabla y_j \mathbf{n} \nabla y_i|}{|\nabla x_i \mathbf{n} \nabla y_j|}, a_{22} = \frac{|\nabla x_i \mathbf{n} \nabla y_j|}{|\nabla x_i \mathbf{n} \nabla y_i|} \end{aligned}$$

A mátrixelemek közös szorzójának négyzete:

$$q = \frac{1}{|\nabla x_i \mathbf{n} \nabla y_i|^2}$$

Ezek felhasználásával a végeredmény a következő formában írható fel:

$$\begin{bmatrix} dx_j \\ dy_j \end{bmatrix} = \mathbf{A} \begin{bmatrix} dx_i \\ dy_i \end{bmatrix} + \frac{q}{2} \begin{bmatrix} dx_i & dy_i \\ -\nabla y_i \cdot \mathbf{P} \cdot \nabla y_i & -\nabla y_i \cdot \mathbf{P} \cdot \nabla x_i \\ -\nabla y_i \cdot \mathbf{P} \cdot \nabla x_i & \nabla x_i \cdot \mathbf{P} \cdot \nabla x_i \\ \nabla y_i \cdot \mathbf{Q} \cdot \nabla y_i & -\nabla y_i \cdot \mathbf{Q} \cdot \nabla x_i \\ -\nabla y_i \cdot \mathbf{Q} \cdot \nabla x_i & \nabla x_i \cdot \mathbf{Q} \cdot \nabla x_i \end{bmatrix} \begin{bmatrix} dx_i \\ dy_i \end{bmatrix}$$

Ahol

$$\mathbf{P} = a_{11}(\mathbf{U}_{\nabla x_i} + \mathbf{T}_{\nabla x_i}) + a_{12}(\mathbf{U}_{\nabla y_i} + \mathbf{T}_{\nabla y_i}) - (\mathbf{U}_{\nabla x_j} + \mathbf{T}_{\nabla x_j})$$

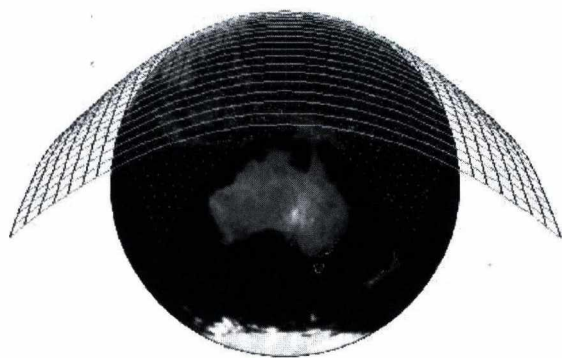
$$\mathbf{Q} = a_{21}(\mathbf{U}_{\nabla x_i} + \mathbf{T}_{\nabla x_i}) + a_{22}(\mathbf{U}_{\nabla y_i} + \mathbf{T}_{\nabla y_i}) - (\mathbf{U}_{\nabla y_j} + \mathbf{T}_{\nabla y_j})$$

Formálisan olyan, mint egy tenzorokra felírt affin transzformáció:

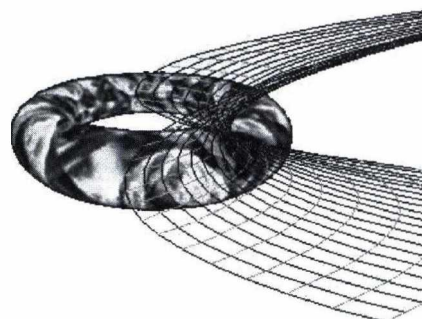
$$\begin{bmatrix} \mathbf{P} \\ \mathbf{Q} \end{bmatrix} = \mathbf{A} \begin{bmatrix} \mathbf{U}_{\nabla x_i} + \mathbf{T}_{\nabla x_i} \\ \mathbf{U}_{\nabla y_i} + \mathbf{T}_{\nabla y_i} \end{bmatrix} - \begin{bmatrix} \mathbf{U}_{\nabla x_j} + \mathbf{T}_{\nabla x_j} \\ \mathbf{U}_{\nabla y_j} + \mathbf{T}_{\nabla y_j} \end{bmatrix}$$

3 EREDMÉNYEK

Az első sorozat a levezetett eredmények helyességét illusztrálja. Az objektumok – tórusz és gömb – implicit módon, előjeles távolság-függvényükkel (signed distance) adottak. Mindkét sorozatban a második kamera az érintősíkkal párhuzamosan lát (azaz az érintősík egy vonalnak látszik). A számítások eredményein megfigyelhető, hogy a választott pontokban az alakzatokat érintő paraboloidok, a gömb esetén forgás-paraboloid (2. ábra), tórusznál paraboloid-nyereg (3. ábra) adódik megoldásként.



2. ábra: gömb közelítése



3. ábra: tórusz közelítése belső nyereg-pontjában

Az alábbi szintetikus képsorozatok az eredményeket mutatják be egy-egy jellegzetes nézetben. A sorozatok első képe az 1. (i.) kamera által mutatott, a második és harmadik a 2. (j.) kamera által mutatott képek. Az eredmények a lineáris és a kvadratikusan közelítések alkalmazhatósági tartományának jelentős eltérését szemléltetik. A harmadik sorozat (tórusz) különösen érdekes abban a tekintetben, hogy az első kép négyzetének kvadratikusan transzformáltja nem egyértékű a másik kamera képén, önmagába visszafordul, azaz a tórusz „hátoldalára” mutat.

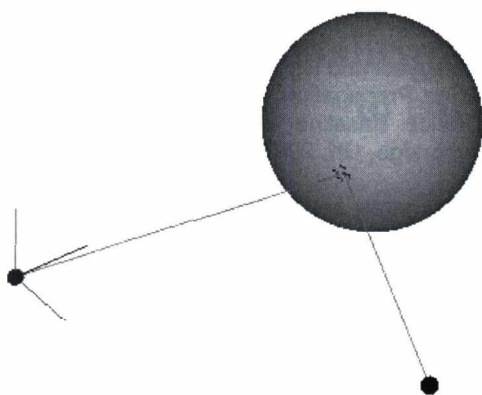
A második kép a 2. kamerán a lineáris, a harmadik a 2. kamerán a kvadratikusan transzformáció eredménye.

4 EGY LEHETSÉGES ALKALMAZÁS

A fenti eredmények egyfajta többkamerás rekonstrukció típus [3] minőségének és megbízhatóságának javítása érdekében születtek. Az említett alkalmazás egyik fő problémája a jobb minőség érdekében alkalmazott nagyobb korrelációs ablakok esetén tapasztalt instabilitás. Várakozásunk szerint ezt a hátrányt sikerül nagymértékben csökkenteni. A tesztek folyamatban vannak, az eredmények megfelelő mennyiségű információ után kerülnek publikálásra.

5 ILLUSZTRÁCIÓK

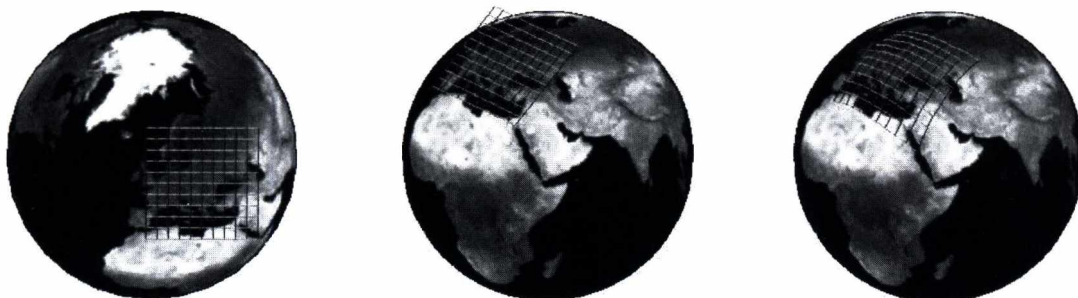
A 4. ábra mutatja be a képsorozatok készítésének körülményeit, ezzel a konfigurációval készült a 2. (gömb) sorozat.



4. ábra: a sorozatok készítésének körülményei: a piros pont az i. kamera, a kék a j. kamera helyzetét, a sárga vonalak az optikai tengely irányát mutatják.

6 ÖSSZEFOGLALÁS

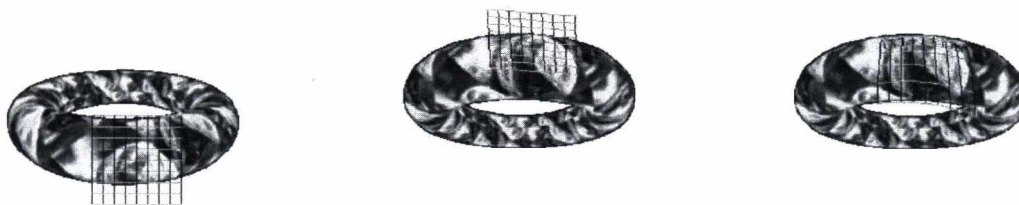
A kvadratikus közelítés szolgáltatja egyenletek nem egyszerűek, de jól programozhatók. A [3] (alap) rekonstrukció és kvadratikus változatának az összehasonlítását szolgáló tesztek éppen csak elkezdődtek. Jelenleg még nem áll rendelkezésre elegendő mennyiségű adat ahhoz, hogy tudjuk: pontosan milyen feltételek mellett (a rekonstrukció „szabadságfoka” igen nagy, a térbeli rácsmérettől a korrelációs ablak méretéig, a felület textúrázottságától a kamerák számáig stb.) milyen mértékű javulás várható a rekonstrukció minőségében. Az első eredmények szerint 20-30%-os javulás várható a megoldás-felülettől való eltérés mértékében. A számítás alapja az ismert (szintetikus) megoldás-felület vertexeinek átlagos távolsága a rekonstrukcióval kapott felülettől.



1. sorozat: Lineáris és kvadratikus közelítés gömbön I.



2. sorozat: Lineáris és kvadratikus közelítés gömbön II.



3. sorozat: Lineáris és kvadratikus közelítés tóruszon

7 REFERENCIÁK

- [1] Z. Zhang
"A flexible new technique for camera calibration" Technical report MSR-TR-98-71, Microsoft Research, 1998,
<http://research.microsoft.com/~zhang/calib/>
- [2] Z. Megyesi and D. Chetverikov
"Affine Propagation for Surface Reconstruction in Wide Baseline Stereo" Proc. ICPR 2004, Cambridge, UK, 2004, vol.4, pp.76-79.
- [3] O. Faugeras, and R. Keriven
"Variational principles, surface evolution, PDE's, level set methods, and the stereo problem" 1998, IEEE Trans. Image Proc., 7(3):336-344.

Discrete orthogonality of Zernike functions and its relevance to corneal topography

A. Soumelidis¹, Z. Fazekas¹, M. Pap² and F. Schipp³

¹ Computer and Automation Research Institute, Budapest, Hungary

² Department of Applied Mathematics, University of Pécs, Pécs, Hungary

³ Department of Numerical Analysis, Eötvös Lóránd University, Budapest, Hungary

Abstract

The optical aberrations of human eyes, as well as, those of other optical systems, are often characterized with the Zernike coefficients. The corneal surface is also frequently represented in terms of the Zernike functions. Although, the corresponding coefficients are normally obtained from measurements at discrete points and via discrete computations, the developers of the corneal measurement devices and evaluation programs could not rely on the discrete orthogonality of the Zernike functions. Recently, meshes of points over the unit disk were found and reported that ensure the discrete orthogonality of the Zernike functions. In the present paper, we report on our experimental results concerning the precision of the Zernike-based surface representation over the unit disk. The test surfaces considered herein include centrally positioned and shifted cones, pyramids, and some cornea-like surfaces.

Categories and Subject Descriptors (according to ACM CCS): F.2.1 [Analysis of algorithms and problem complexity]: Computation of transforms, G.1.2 [Numerical analysis]: Approximation of surfaces and contours, G.1.4 [Numerical analysis]: Error analysis, and I.3.5 [Computer graphics]: Curve, surface, solid, and object representations.

1. Introduction

The optical aberrations of human eyes, as well as, those of other optical systems, are often characterized with the Zernike coefficients. The corneal surface and its aberrations are also frequently represented in terms of the Zernike functions. Although, the Zernike representation of an optical surface or a system is normally obtained from measurements at discrete points and via discrete computations, the developers of the relevant measurement devices and programs could not rely on the discrete orthogonality of the Zernike functions, as it was not known whether it exists, or not. In 2005, however, the discrete orthogonality of the Zernike functions over the unit disk was proven, and appropriate meshes over the unit disk were constructed that ensure this orthogonality⁶. In the present paper, we continue reporting our experimental results concerning the representation and its precision achieved using the aforementioned meshes and the Zernike functions. The achievable precision of the discrete orthogonality has been already looked at via an earlier software

implementation¹². Some initial results concerning the precision of the Zernike-based representation and reconstruction of cornea-like surfaces is in press. In the present paper, we will look particularly at the error functions resulting from the Zernike reconstructions.

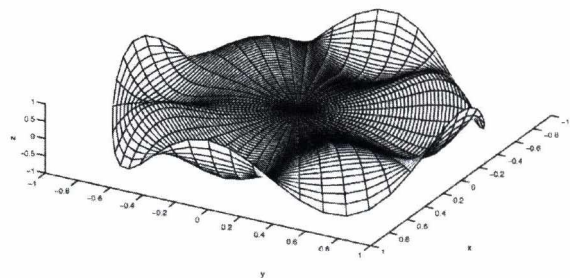


Figure 1: An example of the Zernike functions: Y_3^3 .

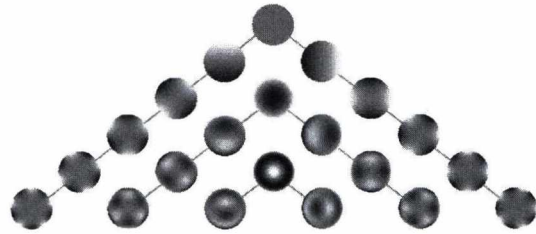


Figure 2: The first twenty-one Zernike functions arranged in a triangle. This arrangement corresponds to the common indexing of the Zernike functions.

In the rest of this introduction, the Zernike functions and their parametrizations are touched upon. Then the significance of these functions in ophthalmology – concerning eye-modelling and eye-related measurements – is pointed out.

In the subsequent sections, we elaborate on how to compute Zernike coefficients in an efficient manner and discuss our experimental results concerning the Zernike-based representations of test surfaces and their precisions.

1.1. The Zernike functions

The orthogonal system of Zernike functions was introduced by Fritz Zernike – a Dutch physicist and Nobel prize winner – to model symmetries and aberrations of optical systems (e.g., telescopes) ¹. In Fig. 1, one of the Zernike functions – namely Y_3^3 – is shown as an example.

In Figs. 2 and 4, small gray-toned renderings of the first few Zernike functions are shown in two different pyramid-like arrangements. The figures show two possible ways of indexing (ordering) the Zernike functions. The indexing represented in Fig. 2 is widely used in the literature, while the indexing represented in Fig. 4 is convenient when dealing with the discretisation proposed in ⁶. The latter indexing of the Zernike functions corresponds to the Y_n^l notation used above. This notation is defined in Section 2.

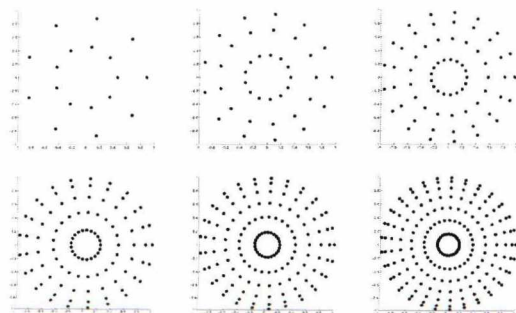


Figure 3: Mesh-dots.

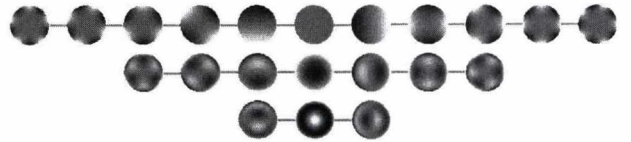


Figure 4: The Zernike functions of Fig. 2 arranged into a trapezoid. This arrangement corresponds to the parametrization used in this paper. The connecting lines placed between images suggest the correspondence between the two index-spaces.

Some of the important and useful properties of the Zernike system are summarized in paper by Pap and Schipp cited above. The mentioned paper can be used as a pointer to a wider range of relevant publications in this topic. In Section 2, the continuous Zernike functions and their indexing used in this paper are presented.

1.2. The Zernike representation used in ophthalmology

Nowadays, the ophthalmologists are quite familiar with the "smoothly waving" Zernike-surfaces, such as shown in Fig. 1. They use these surfaces exactly in the way as was intended by Zernike, that is, to characterize various symmetries and aberrations of an optical system: those of human eyes. In case of corneal topography, the symmetries and the aberrations of the corneal surfaces are examined with – and computationally reconstructed by – corneal topographer devices. In case of wavefront analysis, the optical features of the eye-ball is measured with a Shack-Hartmann wavefront-sensor. These characterizations are given partly in the form of Zernike coefficients. As the optical aberrations may cause serious acuity problems, and are significant factors to be considered in planning of sight-correcting operations, wide range of statistical data – concerning the eyes of various groups of people – is available concerning the most significant Zernike coefficients, see e.g., ⁸. Another inter-

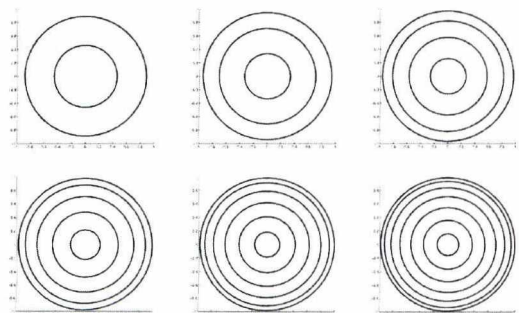


Figure 5: Mesh-circles.

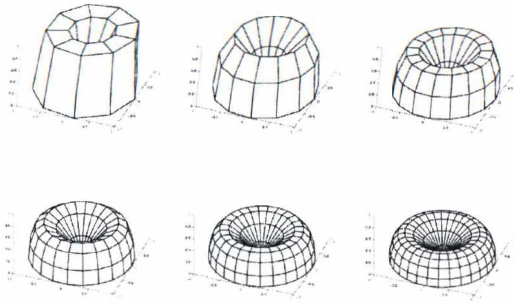


Figure 6: Summation weights – called Christoffel numbers – associated with the individual mesh-points. These weights are shown with interpolation between mesh-points for better presentability.

esting use of the Zernike coefficients was reported recently. The optical aberrations of the eye make it difficult or in certain cases impossible to make *high-resolution retinal images* without compensating these aberrations. However, by properly compensating them, high-resolution retinal imaging can be achieved, as reported in ⁹.

1.2.1. Measuring and modelling corneal surfaces

The purpose of a cornea topographic examination is to determine and display the *shape* and the *optical power* of the *living cornea*. Due to the *high refractive power of the human cornea*, the knowledge of its detailed topography is of great diagnostic importance. The *corneal surface* is often modelled with a *spherical calotte* – see Fig. 13 – though more complex surface models are also available for various purposes, including *testing/calibrating corneal topographers* with non-spherical surfaces ⁷. Cornea-like test surfaces are shown in Figs. 13, 15 and 17.

Recent *eye models* are more realistic and more detailed – certain models, for instance, incorporate even the slight continuous change of the refractive index within the crystalline lens – and rely on *several tuneable parameters*. One of the motivations for using more realistic eye models is their capability to estimate – the actual and the achievable – the *retinal image quality*. The inclusion of fine details into the eye model, however, does not prejudice the optical importance of the anterior surface of the cornea ¹⁰.

The *monocular reflection-based cornea topographers* evaluate the virtual image of some *measurement pattern* that is reflected by and after reflection somewhat *distorted* by the corneal surface. Many of these topographers use systems of *bright and dark concentric rings* – called *Placido disks* – as measurement-patterns. The Placido disks look somewhat similar to the binary patterns appearing in Fig. 14. The surface reconstruction method used in conjunction with the conventional *Placido-based topographers* is rather

problematic, as no point-to-point correspondences are available for the purpose of surface reconstruction ⁴. On the other hand, monocular topographers with *more distinguishable measurement patterns* are still popular. In the recent years, several *multi-camera corneal measurement devices and methods* were proposed and implemented, e.g., ¹¹.

1.2.2. Relevance of discrete orthogonality

Although, Zernike coefficients were obtained from *measurements at discrete corneal points* and via discrete computations, the developers of the corneal measurement devices and shape-evaluation programs could not rely on the discrete orthogonality – see e.g., ⁵ – Zernike functions simply because no mesh of points ensuring discrete orthogonality was known. Not surprisingly, the discrete orthogonality of Zernike functions was a *target of research* for some time, e.g., ³. Only recently were meshes of points – ensuring discrete orthogonality of the Zernike functions – found and reported in ⁶.

The meshes introduced there are used herein to calculate the Zernike-based representations and their precisions for some test surfaces. These test surfaces include three “*cornea-like*” test surfaces, as well. In Section 3, the *discretization approach* introduced in ⁶ is presented briefly. In Section 4, an *efficient way of computing the Zernike coefficients* is presented. In Section 5, the *concrete Zernike-based surface representations* and their precisions are discussed. We conclude in Section 6.

2. The continuous Zernike functions

A surface over the *unit disk* can be described by a two-variable function $g(x, y)$. The application of the polar-transform to variables x and y results in

$$x = \rho \cos \vartheta, \quad y = \rho \sin \vartheta, \quad (1)$$

where ρ and ϑ are the *radial* and the *azimuthal* variables, respectively, over the unit disk, i.e., where

$$0 \leq \rho \leq 1, \quad 0 \leq \vartheta \leq 2\pi. \quad (2)$$

Using ρ and ϑ , $g(x, y)$ can be *transcribed* in the following form:

$$G(\rho, \vartheta) := g(\rho \cos \vartheta, \rho \sin \vartheta). \quad (3)$$

The set of Zernike polynomials of degree less than $2N$ is as follows.

$$Y_n^l(\rho, \vartheta) := \sqrt{2n + |l| + 1} \cdot R_{|l|+2n}^{|l|}(\rho) \cdot e^{il\vartheta}, \\ (l \in \mathbb{Z}, \quad n \in \mathbb{N}, \quad |l| + 2n < 2N)$$

The *radial Zernike polynomials* $R_{|l|+2n}^{|l|}$ can be expressed with the *Jacobi polynomials* $P_k^{\alpha, \beta}$ in the following manner:

$$R_{|l|+2n}^{|l|}(\rho) = \rho^{|l|} \cdot P_n^{0,|l|}(2\rho^2 - 1). \tag{4}$$

3. Discretization of Zernike functions

The mesh, i.e., the set of nodal points – given in ⁶ and proven to ensure the discrete orthogonality of Zernike functions over this mesh – is as follows:

$$X_N := \{z_{jk} := (\rho_k^N, \frac{2\pi j}{4N+1}) : k = 1, \dots, N, j = 0, \dots, 4N\}, \tag{5}$$

where

$$\rho_k^N := \sqrt{\frac{1 + \lambda_k^N}{2}}, \quad k = 1, \dots, N, \tag{6}$$

and where λ_k^N is the k -th root ($k = 1, \dots, N$) of the Legendre polynomial P_N of order N . In Fig. 3, meshes X_2, X_3, \dots, X_7 are shown as concrete examples; the corresponding circular meshes are shown in Fig. 5 for comparison purposes.

By using the discrete integral of (7), the discrete orthogonality of the Zernike functions was proven in ⁶. The discrete orthogonality relation is given in (8).

$$\int_{X_N} f(\rho, \varphi) dV_N := \sum_{k=1}^N \sum_{j=0}^{4N} f(\rho_k^N, \frac{2\pi j}{4N+1}) \frac{A_k^N}{2(4N+1)} \tag{7}$$

In (7), the A_k^N 's are the weights that are associated with the discrete circular rings in the mesh. See Fig. 6. These weights – called Christoffel numbers – are derived for the radial Zernike polynomials from the quadrature formula of Legendre polynomials P_N of order N via the application of the argument transform used in (4).

$$\int_X Y_n^m(\rho, \varphi) \overline{Y_{n'}^{m'}(\rho, \varphi)} dV_N = \delta_{nn'} \delta_{mm'}. \tag{8}$$

In the above orthogonality relation, $n + n' + |m| < 2N$ and $n + n' + |m'| < 2N$ are assumed to be satisfied.

The quadrature formulas are significant tools in constructing discrete orthogonal systems. Quadrature formulas are known for some well-researched continuous orthogonal polynomials – of certain importance – of one variable since Gauss's time. See e.g., ². The quadrature formulas are expressed in the following way:

$$\int_{-1}^1 f(x) dx \approx \sum_{k=1}^N f(\lambda_k^N) A_k^N. \tag{9}$$

Interestingly, the integration of function $f(x)$ via the quadrature formulas is much more precise than the numerical integration over some arbitrary (e.g., equidistant mesh).

In our case, that is, for the discretization of the radial Zernike polynomials – i.e., the radial component of the Zernike functions – the N roots of Legendre polynomials P_N were used. The exact formula for deriving the A_k^N weights is not given here, for this see ⁶. We just note here that the formula is exact for every polynomial f of order less than $2N$.

For $N = 2, \dots, 7$, the aforementioned A_k^N weights – namely, $A_1^2, A_2^2; A_1^3, A_2^3, A_3^3; \dots; A_1^7, \dots, A_7^7$ – are shown over their corresponding X_N discrete meshes in Fig. 6.

4. Computing the discrete Zernike coefficients

The discrete Zernike coefficients associated with function $T(\rho, \varphi)$ can be calculated with the following discrete integral:

$$C_n^m = \frac{1}{\pi} \int_{X_N} T(\rho, \varphi) \overline{Y_n^m(\rho, \varphi)} dV_N. \tag{10}$$

It is worth noting that if $T(\rho, \varphi)$ happens to be an arbitrary linear combination of Zernike functions of degree less than $2N$, then the above discrete integrals, i.e., for n 's and m 's satisfying the inequality $2n + |m| < 2N$, result in the exact Zernike coefficients, i.e., which are calculated from the corresponding continuous integrals. The computing of the discrete Zernike coefficients – according to (10) – was speeded up via using FFT. It is made possible by the presence of the expression appearing between parentheses in the last line of (11), which is the discrete Fourier transform of T over the $4N + 1$ points of the discrete circle – see Figs. 3 and 5 – with radius ρ_k^N .

$$\begin{aligned} & \frac{1}{\sqrt{2n + |l| + 1}} \int_{X_N} T(\rho, \varphi) \overline{Y_n^m(\rho, \varphi)} dV_N = \\ & = \sum_{k=1}^N \sum_{j=0}^{4N} T(\rho_k^N, \frac{2\pi j}{4N+1}) R_{2n+|m|}^{|m|}(\rho_k^N) \frac{A_k^N}{2(4N+1)} e^{-i \frac{2\pi j}{4N+1}} = \\ & = \frac{1}{2(4N+1)} \sum_{k=1}^N A_k^N R_{2n+|m|}^{|l|} \left(\sum_{j=0}^{4N} T(\rho_k^N, \frac{2\pi j}{4N+1}) e^{-i \frac{2\pi j}{4N+1}} \right) \end{aligned} \tag{11}$$

In the discussion of the concrete computational results presented in Figs. 7-18, we will refer to the continuous counterparts – see Fig. 5 – of the discrete circles as sampling circles, and the cylinders over them – appearing in Figs. 7, 9, 11, 13, 15 and 17 – as sampling cylinders.

5. Zernike representation of some test surfaces

In this section, we look at the Zernike-based representation and reconstruction of some test surfaces over the unit disk. MATLAB routines were created for computing the surface points of these test surfaces and their discrete Zernike representations.

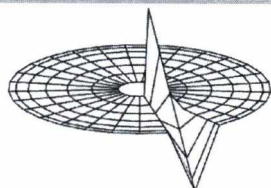
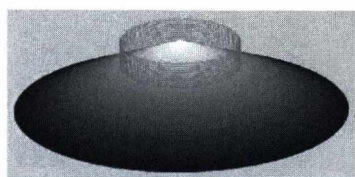


Figure 7: (a) The intersection of a "sampling" cylinder and a centrally positioned cone. (b) The circular Fourier transforms of the above cone. (c) The Zernike coefficients of the cone.

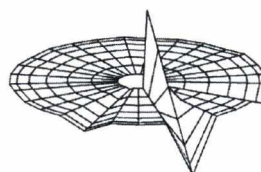
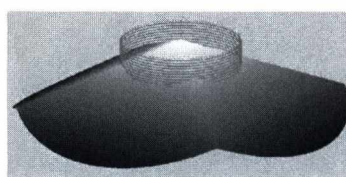


Figure 9: (a) The intersection of a "sampling" cylinder and a central pyramid over the unit disk. (b) The circular Fourier transforms of the pyramid. (c) The Zernike coefficients of the pyramid.

The surfaces included in this study are as follows: cone (Fig. 7), pyramid (Fig. 9), a shifted cone (Fig. 11), a sperical calotte (Fig. 13), a shifted spherical calotte (Fig. 15) and a sphero-cylindrical surface (Fig. 17). Clearly, the first three are rather artificial surfaces, they have nothing to do with the corneal shapes. They serve as easy-to-grasp examples

that can be easily visualized and their interaction with the sampling mesh can be easily followed. The rest of the test surfaces were selected from the ones proposed in⁷. These are more cornea-like, and often used in cornea modelling and in characterizing the shape of the human cornea.

For each of the above test surfaces, (a) the original sur-

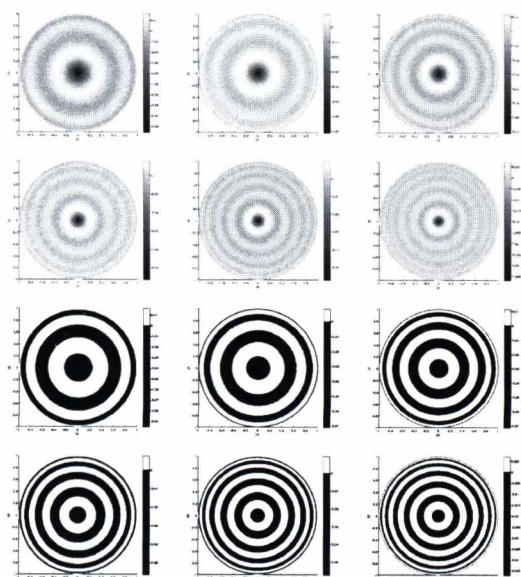


Figure 8: Reconstruction errors at various parameter-settings for a cone.

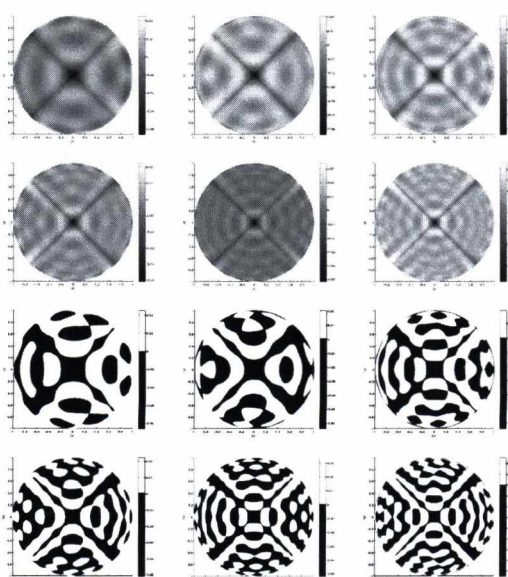


Figure 10: Reconstruction errors at various parameter-settings for a pyramid.

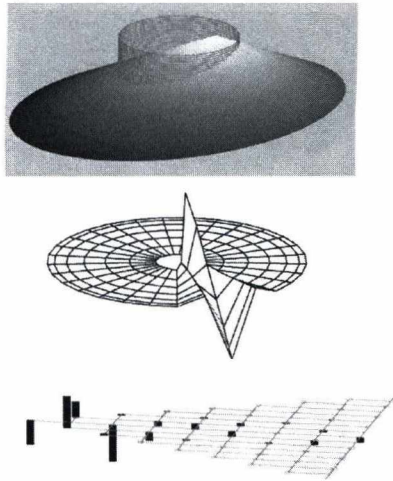


Figure 11: (a) The intersection of a "sampling" cylinder and a non-centrally positioned cone. (b) The circular Fourier transforms of the non-centrally positioned cone. (c) The Zernike coefficients of the non-centrally positioned cone.

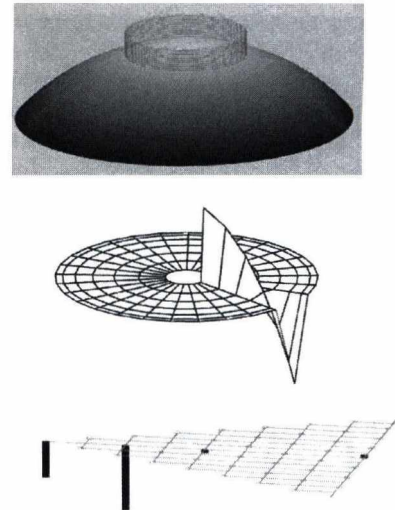


Figure 13: (a) The intersection of a "sampling" cylinder and a centrally positioned "cornea-like" spherical calotte. (b) Its circular Fourier transforms. (c) The Zernike coefficients of the spherical calotte.

face rendered in gray tone and viewed from a skew angle; (b) the intersection of the surface with one of the sampling cylinders (i.e., a cylinder over one of the discrete circles of the mesh); (c) the discrete Fourier spectrum of the surface points sampled by the discrete circles of the mesh; (d) the discrete Zernike spectrum of the surface with the Zernike-

coefficients arranged according to pyramid shown in Fig. 2; (e) the signed error functions corresponding to Zernike-based reconstructions over increasingly dense X_N meshes, and (f) their thresholded (threshold = 0, i.e., at locations marked with black the original surface was above the recon-

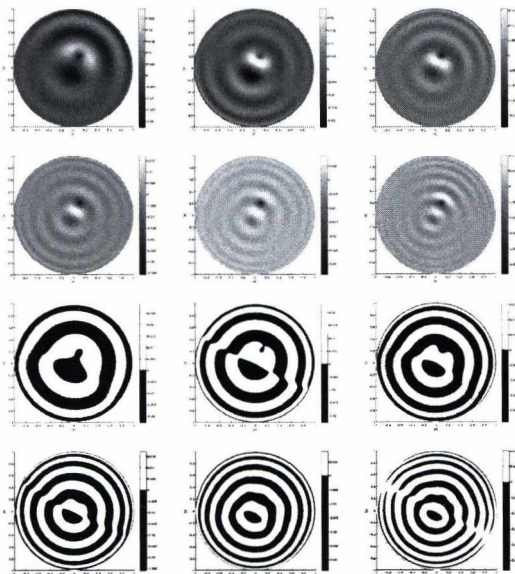


Figure 12: Reconstruction errors at various parameter-settings for a non-centrally positioned cone.

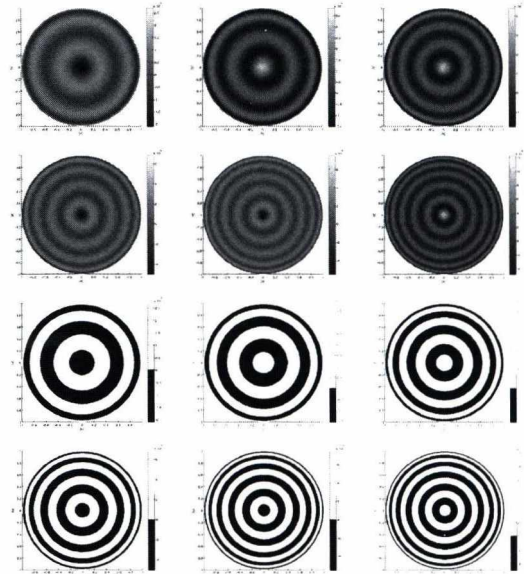


Figure 14: Reconstruction errors at various parameter-settings for a central spherical calotte.

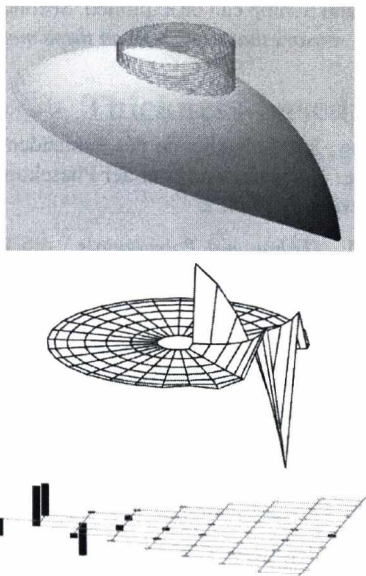


Figure 15: (a) The intersection of a "sampling" cylinder and a non-centrally positioned spherical calotte. (b) Its circular Fourier transforms. (c) The Zernike coefficients of the non-centrally positioned spherical calotte.

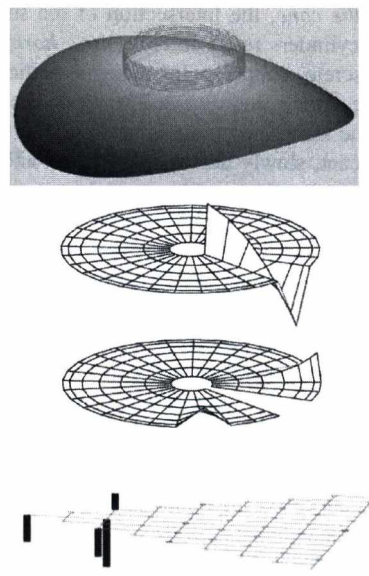


Figure 17: (a) The intersection of a "sampling" cylinder and a centrally positioned sphero-cylindrical surface. (b) Its circular Fourier transforms scaled up. (c) The Zernike coefficients of the sphero-cylindrical surface.

structed one, at locations marked with white it was the other

way round) – "zebra-like" – versions are presented. See Figs. 7-18.

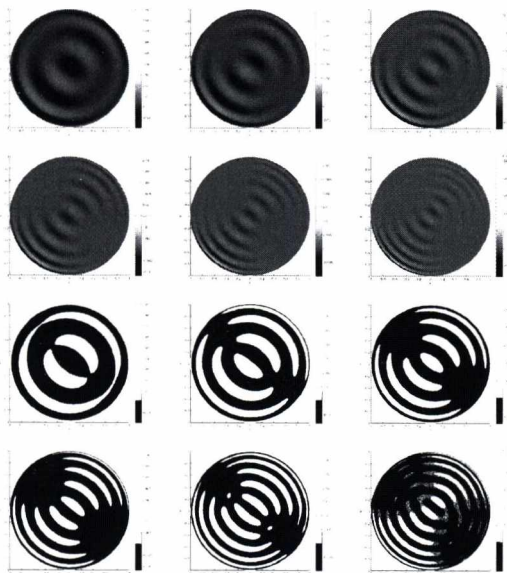


Figure 16: Reconstruction errors – presented as in gray tone and binary images – at various parameter-settings for a non-centrally positioned spherical calotte.

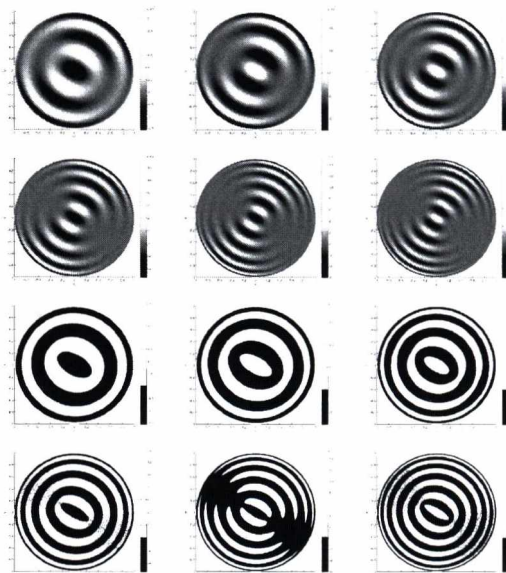


Figure 18: Reconstruction errors at various parameter-settings for a centrally positioned sphero-cylindrical surface.

In case of the cone, the intersection of the surface and the sampling cylinders result in *concentric horizontal circles*. Each discrete sample point along a particular such circle have the same z-coordinate. Therefore, the sampling achieved by the sampling points is perfect. There is, however, a significant, slowly decreasing error at and near the *apex of the cone*, as there are no sample points available there (due to the particular choice of the mesh). As the z-coordinates of the surface points are constant above a particular sampling circle, the *discrete Fourier spectrum* exhibits only a *single peak at zero frequency* for that circle. The zero-frequency Fourier coefficients form a *straight – i.e., linear – ridge* in the *discrete Fourier-spectra* (see Fig. 7b).

In many respects the situation is similar in case of the *spherical calotte*, the intersection of the surface and the sampling cylinders are again *concentric horizontal circles*; along existing sampling circles the sampling is again perfect. However, there is no significant error over the origin as the calotte surface is not pointed. The *discrete Fourier spectra* exhibit *peaks only at zero frequency*. The shape of the resulting *ridge* is now forms an *ellipse arc* (or a *circular arc* after proper scaling). Some of the zebra-ring error patterns appear negated in this case.

The case of the *pyramid* is in many respects similar to that of the cone, however, there are interesting dissimilarities. Firstly, the intersection of the pyramid and the sampling cylinder is no longer a circle. It is *not even a planar curve*. The z-coordinates of the surface points above a particular sampling circle are no longer constant. The *spatial curve* of the intersection is not everywhere differentiable. The sampling is far from being perfect above the discrete circles and cause representation errors resulting in the distortion of the error patterns (see Fig. 10). It is also interesting to see that the *error-patterns are no longer symmetric*. (Though, the pyramid itself is invariant on $k\frac{\pi}{2}$ rotations, the sampling mesh is not.) The other cases, are also interesting and it would be worthwhile to go through them one by one. It, however, has to be left to the interested reader.

6. Conclusion

The discretization scheme used in this paper has relevance to the concrete application field, i.e., corneal measurements, but could also benefit physicists and engineers dealing with *optical measurements*, or with the *design of optical measurement devices*. In our *experiments* concerning the *Zernike-based representation* and reconstruction of surfaces, we looked at the *structure of the error functions* and their thresholded binary versions ("*zebra rings*"). The *structure of these rings* was discussed in some detail. *Further work* is going on – concerning the mentioned meshes and the discrete Zernike system – in *two main directions*. Firstly, how to *tune the conventional measurements* – i.e., the measurement based on other meshes – so that the advantages of the meshes pro-

posed by Pap and Schipp can be exploited. Secondly, how to *design optical sensors that are based on these meshes*.

References

1. F. Zernike. Beugungstheorie des Schneidenverfahrens und seiner verbesserten Form der Phasenkontrastmethode. *Physica*, **1**, 1934. 2
2. G. Szegő. *Orthogonal Polynomials*. 4th Ed., AMS, New York, NY, USA, 1981. 4
3. J. C. Wyant and K. Creath. *Basic Wavefront Aberration Theory for Optical Metrology*. in *Applied Optics and Optical Engineering*, Vol. 11, Academic Press, New York, NY, USA, 1992. 3
4. M. C. Corbett, E. S. Rosen, and D. P. S. O'Brart. *Corneal Topography: Principles and Applications*. Bmj Publ. Group, London, UK, 1999. 3
5. D. R. Iskander, M. J. Collins and B. Davis. Optimal Modeling of Corneal Surfaces with Zernike Polynomials. *IEEE Transactions on Biomedical Engineering*, **48**(1), pp. 87–95, 2001. 3
6. M. Pap and F. Schipp. Discrete Orthogonality of Zernike Functions. *Mathematica Pannonica*, **16**(1), pp. 137–144, 2005. 1, 2, 3, 4
7. A. Soumelidis and B. Csákány. *Specification of Test Corneal Surfaces*. Project Report (CORNEA-INT-2M02), MTA SZTAKI, Budapest, Hungary, 2005. 3, 5
8. T. O. Salmon and C. van de Pol. Normal-eye Zernike Coefficients and Root-mean-square Wavefront Errors. *Journal of Cataract & Refractive Surgery*, **32**(12), 2006. 2
9. N. Ling, Y. Zhang, X. Rao, C. Wang, Y. Hu, and C. Jiang. *Adaptive Optical System for Retina Imaging Approaches Clinic Applications*. in *Series Springer Proceedings in Physics*, Springer Verlag, Berlin, Germany, pp. 2064–2074, 2006. 3
10. M. S. de Almeida and L. A. Carvalho. Different Schematic Eyes and Their Accuracy to the In Vivo Eye: a Quantitative Comparison Study. *Brazilian Journal of Physics*, **37**(2A), pp. 378–387, 2007. 3
11. Z. Fazekas, A. Soumelidis, A. Bódis-Szomorú, and F. Schipp. "Specular Surface Reconstruction for Multi-camera Corneal Topographer Arrangements", *30th Annual International Conference of IEEE EMBS*, Vancouver, Canada, pp. 2254–2257, (2008). 3
12. Z. Fazekas, A. Soumelidis and F. Schipp. "Utilizing the Discrete Orthogonality of Zernike Functions in Corneal Measurements", *World Congress on Engineering*, International Association of Engineers, London, UK, pp. 795–800 (2009). 1

Thickness-based binary morphological improvement of distorted digital line intersections

H. Tomán¹, A. Hajdu¹, J. Szakács¹, D. Hornyik¹, A. Csutak² and T. Pető³

¹Faculty of Informatics, University of Debrecen, 4010 Debrecen, POB 12, Hungary

²Medical and Health Science Center, University of Debrecen, Nagyerdei Krt. 98, 4032 Debrecen, Hungary

³Moorfields Eye Hospital, 162 City Road, London EC1V 2PD, United Kingdom

Abstract

This paper describes a discrete geometrical approach to improve the quality of line/curve intersections in the skeletonized image. A proper geometrical foundation is presented to be able to measure the degree of the distortion of the intersections caused by skeletonization. An improvement to reduce this distortion is suggested based on the separation of thick and thin lines/curves in the original image. This step assures the appropriate skeletonization of the thick elements which is consequently connected by the skeletons of the thin elements by direction estimation. The approach can be applied in improving the skeletonization of the retinal vascular system which step is often considered in the detection of several vascular diseases of the eye. Corresponding experimental results are also reported based on a publically available database.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 Computational Geometry and Object Modeling

1. Introduction

Skeletonization¹ is a binary morphological operation to extract the centerline of an object. The skeleton is often found by thinning the object with removing pixels without affecting the general shape. The skeleton is a popular object representation, since pixelwise observations can be used to characterize the spatial behavior of the objects. For example, junctions can be located as pixels with at least three 8-neighbors. However, the skeleton is usually distorted as it is illustrated in Figure 1, with having two junction pixels instead of a single one. In such cases, the intersecting elements cannot be tracked properly later on.

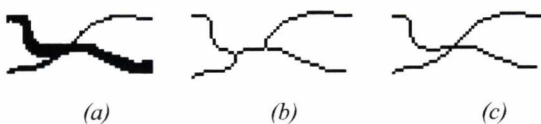


Figure 1: Distortion of an intersection during skeletonization; (a) original image, (b) its skeleton, (c) desired skeleton.

Several approaches have been released to overcome this problem. Basically, these methods can be grouped as skeleton-based or global approaches. In the skeleton-based case², the distorted locations are found at the already skeletonized images, while in the global case the intersections are tried to be localized in the original line drawing image. The usual drawback of a skeleton-based decision is that it needs a proximity parameter to differentiate between true intersections and enclosed segments connecting intersection points and the definition of this parameter is challenging for the whole image. Global approaches usually based on corner detection^{3,4} inheriting the typical related problems, like too many/few corner candidates. However, their strength is that they try to locate and improve distorted intersections without touching the original image.

We have encountered with this intersection distortion problem during the improvement of the skeleton of the retinal vascular system. As a specific field, in automatic screening of retinopathy the proper mapping of the vasculature system has very important role in gathering information to diagnose diseases. For example, a proper traversal of the vessels gives information about how the thickness of the vessels changes, or about the artery/vein ratio. After applying a segmentation method⁵ a binary image can be

extracted with similarities to line drawing ones (see Figure 2).



Figure 2: Binary vessel map of the retina.

In this paper, we propose a global technique for suppressing distorted intersections in the skeleton that can be considered as both a stand-alone or a complementary approach to others e.g. mentioned before. Our method is based on the separation of the thick and thin vessels of the complete vascular system (input image) before applying skeletonization. The main motivation behind this approach is to be able to extract the precise skeletonization of the thick vessels at the intersection, while the skeletonization of the thin components is executed separately. However, since the vessel system is split into two parts a consecutive reconnecting step is needed to connect the thin skeletal elements to the thick ones. The rest of the paper is organized as follows. Section 2 describes the theoretical model that is considered to measure the distortion as a function of the thickness of the intersecting elements. In section 3 the separation of the thick/thin elements is explained together with the reconnecting step to connect their skeletons. Section 4 contains our experimental results. Some pending issues are discussed and conclusions are drawn in section 5.

2. Theoretical model for vessel intersections

As the theoretical model for the intersection of two vessels, we consider two stripes bounded by pairs of parallel lines for the intersecting vessels as shown in Figure 3.

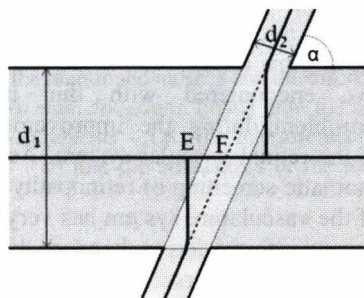


Figure 3: Geometric model for the intersection of two vessels.

For simplicity, we perform the detailed geometric calculation for only one half of the intersection since the other half distorts in the same way because of geometric symmetry. To measure the degree of the distortion of the intersection, a natural error function can be defined as the distance between the desired and actual junction points (see F and E in Figure 3, respectively).

For the calculation our coordinate system is selected to have one of the line pairs is perpendicular to the axis x while the pitch of the other one with the axis x is α . The distances of the parallel lines, that is, the widths of the stripes, are denoted by d_1 and d_2 , where $d_1 \leq d_2$. This model lets us to measure the degree of the distortion of the skeletal intersection as a function of the width of the stripes and their enclosed angle. For defining the skeleton, we follow the classical definition¹ that builds it up from centers of the circles of maximum radius among all circles which lie within the object.

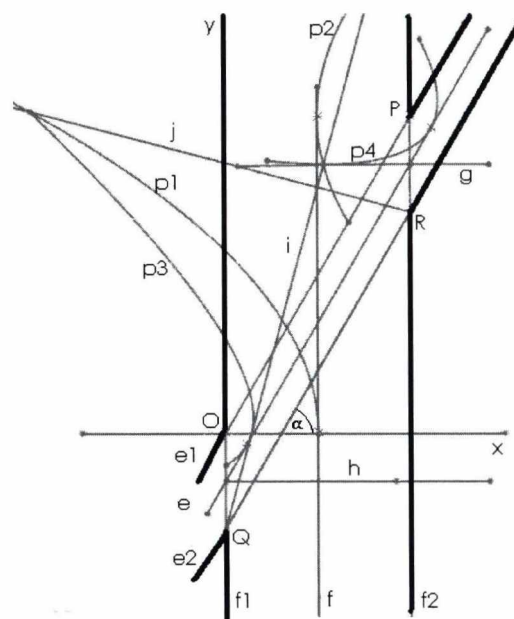


Figure 4: Geometric elements defining the skeleton of the model.

Based on Figure 4, we can recognize the following geometrical components that take part in building up the skeleton for our intersection model:

- e : the centerline of the stripe bounded by (e_1, e_2) ,
- f : the centerline of the stripe bounded by (f_1, f_2) ,
- p_1, p_2, p_3, p_4 : the respective parabolas given by focuses and directrices (O, f_2) , (P, f_1) , (O, e_2) and (P, e_1) ,
- g, h : the respective perpendicular bisector of the line segments PR and OQ ,
- i, j : the respective angle bisectors of e_2 and f_1 at Q , e_1 and f_2 at R .

The parameters of the model are:

- d_1, d_2 : the respective widths of the intersecting stripes bounded by (e_1, e_2) and (f_1, f_2) ,
- k : the d_2/d_1 ratio which will be very useful in simplifying some calculations and also in visualizing the degree of distortion based on the relative widths of the stripes,
- α : the enclosed angle between the stripe e and the axis x .

The desired skeleton should be found as the intersection point of centerlines of the stripes, whose equation can be given as:

e :
$$y = x \tan \alpha - \frac{d_1}{2 \cos \alpha},$$

f :
$$x = \frac{d_2}{2},$$

respectively. The respective equations of the perpendicular bisectors g and h can be calculated as:

g :
$$y = d_2 \tan \alpha - \frac{d_1}{\cos \alpha},$$

h :
$$y = -\frac{d_1}{2 \cos \alpha}.$$

The respective equations of the angle bisectors i and j can be calculated as:

i :
$$y = \tan\left(\frac{\pi}{4} + \frac{\alpha}{2}\right)x - \frac{d_1}{\cos \alpha},$$

j :
$$y = \tan\left(\frac{3\pi}{4} + \frac{\alpha}{2}\right)(x - d_2) + d_2 \tan \alpha - \frac{d_1}{\cos \alpha}.$$

The respective equations of the parabolas p_1, p_2, p_3 and p_4 can be given by:

p_1 :
$$-x = \frac{1}{2d_2}y^2 - \frac{d_2}{2},$$

p_2 :
$$x = \frac{1}{2d_2}(y - d_2 \tan \alpha)^2 + \frac{d_2}{2},$$

p_3 :
$$y' = \frac{1}{2d_1}x'^2 - \frac{d_1}{2},$$

p_4 :
$$y' = \frac{1}{2d_1}(x' - d_2)^2 + d_2 \tan \alpha - \frac{d_1}{2},$$

where

$$x' = x \cos \alpha - y \sin \alpha,$$

$$y' = x \sin \alpha + y \cos \alpha.$$

As it can be easily checked, the angle bisectors i and j and the centerline f intersect at a single point.

This point is the center of a circle with radius $\frac{d_2}{2}$. It

also follows from the geometric construction that the two parabolas and an appropriate angle bisector (p_1, p_3, j and p_2, p_4, i) intersect in a single point.

In Figure 4, the geometric construction of the skeleton points can be seen with parameters $d_1 = 10$,

$$d_2 = 40 \text{ and } \alpha = \frac{\pi}{3}.$$

It depends on the model parameters which intersection point of the appropriate geometric objects will define the location of the distorted junction point. In the following two sections we will discuss the two possible cases. The first case is mainly valid for large values of k , while the second one is for small values of k . The proper final error term for the degree of distortion is derived from these two cases conditionally to the model parameters in section 2.3.

2.1. Junction point based on bisectors

The intersection point E_1 of the centerline f and the perpendicular bisector g can be calculated as:

$$x_1 = \frac{d_2}{2}, \quad y_1 = d_2 \tan \alpha - \frac{d_1}{2 \cos \alpha}.$$

Performing skeletonization of the two intersecting vessels separately the desired intersection point F has the coordinates:

$$x_0 = \frac{d_2}{2} \text{ and } y_0 = \frac{d_2 \tan \alpha}{2} - \frac{d_1}{2 \cos \alpha}.$$

In this case, the degree of distortion DD_1 is calculated as:

$$DD_1 = \sqrt{(x_1 - x_0)^2 + (y_1 - y_0)^2} = \frac{d_2 \tan \alpha}{2} = \frac{k d_1 \tan \alpha}{2}$$

where $k = \frac{d_2}{d_1}$.

Figure 5 shows the behavior of the error function DD_1

for $\alpha \left(0 \leq \alpha < \frac{\pi}{2}\right)$ and $k \left(1 \leq k \leq 100\right)$ if $d_1 = 1$.

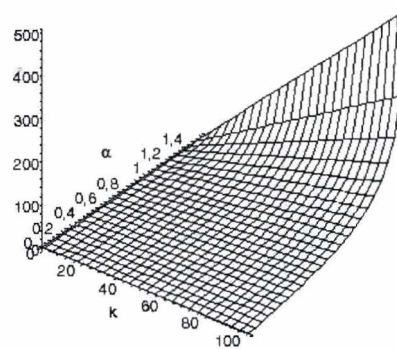


Figure 5: The degree of distortion DD_1 in the first case.

It suggests that the distance of the intersection points changes linearly with the variable k if the value of the other parameter α and d_1 is given. It can be seen also on the plot that the error function is almost

linear in variable α if $0 \leq \alpha < \frac{\pi}{3}$ but it increases rapidly, when $\frac{\pi}{3} \leq \alpha < \frac{\pi}{2}$.

2.2. Junction point based on parabolas

In this case, the intersection point of the parabolas and the corresponding angle bisector that we consider as a skeleton point is $E_2(x_2, y_2)$, where y_2 is the smaller root of the quadratic equation

$$-\cos^2 \alpha \tan\left(\frac{\pi}{4} + \frac{\alpha}{2}\right)x^2 + \left(2d_2 \cos \alpha \left(\sin \alpha \tan\left(\frac{\pi}{4} + \frac{\alpha}{2}\right) + \cos \alpha\right)\right)x + 2d_1 d_2 \cos \alpha - d_2^2 \tan\left(\frac{\pi}{4} + \frac{\alpha}{2}\right) = 0$$

and

$$x_2 = \frac{1}{\tan\left(\frac{\pi}{4} + \frac{\alpha}{2}\right)} \left(y_2 + \frac{d_1}{\cos \alpha}\right).$$

The discriminator of the quadratic equation determining y_2 is always non-negative for $k \geq 1$, and $0 \leq \alpha < \frac{\pi}{2}$. We omit the complex formula for y_2 .

In this case, the error of the skeletonization is the distance of the point $E_2(x_2, y_2)$ from the ideally detected intersection point F . Thus, now we have

$$DD_2 = \sqrt{(x_2 - x_0)^2 + (y_2 - y_0)^2}.$$

With the investigation of the error function we get that the distance of the intersection points depends on the thickness (of the vessels) linearly which can be seen easily from the geometric construction as well. We can consider the error of the skeletonization as a function of the variable α and k .

Figure 6 shows the behavior of the error function with variable α ($0 \leq \alpha < \frac{\pi}{2}$) and variable k ($1 \leq k \leq 100$) if $d_1 = 1$.

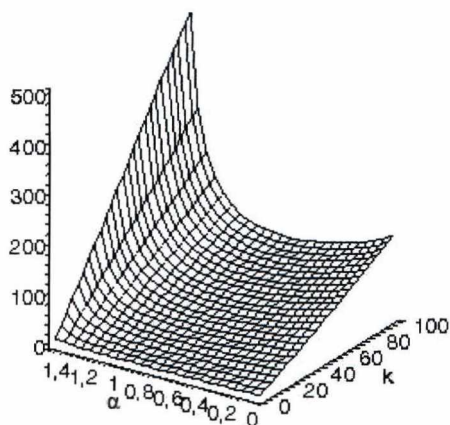


Figure 6: The degree of distortion DD_2 in the second case.

2.3. The total degree of the distortion

After determining the distortion junction point E_1 and E_2 , the total degree of the distortion is decided by the two candidates depending on the parameter k and α . The total degree of the distortion DD is either DD_1 in case the distortion junction point is E_1 or DD_2 if the distortion junction point is E_2 .

It can be seen that $DD_1 \leq DD_2$ for all values of parameter k and α .

It follows from the geometric construction that

$$d(E_2, P) = d(E_2, e_2) = d(E_2, f_1) > \frac{\max(d_1, d_2)}{2} = \frac{d_2}{2}$$

for the radius of the maximal circle with centre point E_2 . It is required in the classical definition of the skeleton that the circle with maximal radius to remain inside the object. Considering this condition the radius $d(E_2, P)$ has to satisfy:

$$d(E_2, P) < \max(d_1, d_2) = d_2$$

Summing up, the total degree of the distortion DD is DD_2 if

$$\frac{d_2}{2} < d(E_2, P) < d_2,$$

otherwise the DD can be considered as DD_1 .

The distance $d(E_2, P)$ depends on the parameter d_1 linearly so the decision about the candidate junction points can be made just by the parameter k and α (see Figure 7).

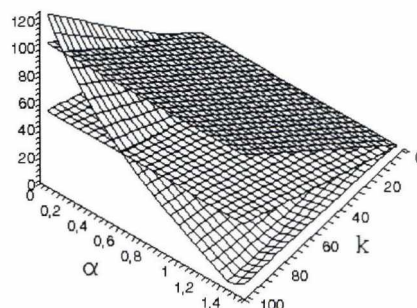


Figure 7: The distance function of E_2 and P , that defines the total degree of the distortion.

We can see on Figure 5 and 6 that the degree of distortion is large for stripes having large width differences. This observation motivated the separation of the vessels based on their widths as it will be discussed in the following section.

3. Improving the skeleton of intersections

Our approach to improve the skeleton of binary vascular images is based on two major steps. First, we split the vascular system into two parts containing the thick and thin vessel segments and perform the

skeleton of the two sets. Then, we reconnect the components disconnected in the splitting step.

3.1. Splitting the vascular system into two parts

The separation of the thick and thin vessels of the whole vascular system is performed by applying erosion steps that remove pixels having less than N 8-neighboring pixels. We repeat the procedure recursively until no more changes are found in the image. By this process, thin blood vessels are erased with the thick vessel subsystem preserved. Then, the thin vascular subsystem can be trivially generated by subtracting the thick one from the original binary image. The result of this splitting procedure is shown in Figure 8 for the input binary vascular system depicted in Figure 2.

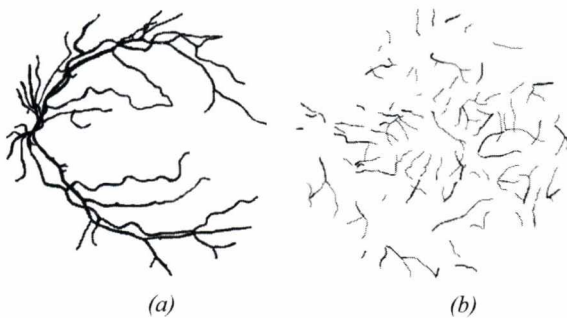


Figure 8: The result of the splitting step; (a) the thick vessels subsystem, (b) the thin vessels subsystem.

We note that our approach for erosion is slightly different from the classical morphological operation. Namely, our approach preserves all the vessels above a given width to guarantee that the thick vessels remain connected.

After successfully splitting the vascular system into two parts, we perform the skeletonization of the thick and thin subsystems. For this purpose, any skeletonization algorithm can be used. In our implementations we considered the one recommended by Deutsch⁶.

3.2. Reconnecting thick and thin elements

When taking the union of the skeletons of the thick and thin vessel subsystems, several discontinuities (gaps) remain to be filled in. This phenomenon is also depicted in Figure 9.

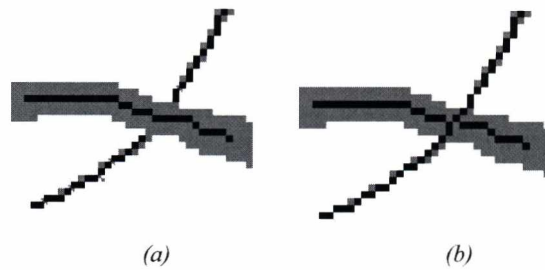


Figure 9: The reconnection step to fill in the gaps between the skeletons of the thick and thin vessels; (a) a gap to be filled in, (b) result after reconnecting the separate skeletons.

To fill in such gaps, we perform direction estimations at the corresponding endpoints of the thin skeleton to connect them to the thick skeleton subsystem. The direction estimation is performed by following the classical recommendation of discrete geometry⁷ for the calculation of a tangent at the given endpoint. That is, the reconnecting line is calculated from the endpoint and the pixel on the skeleton of thin vessel having P -pixel-distance from the endpoint. The value of P is reduced iteratively from a threshold until a successful reconnection is found. There is a wide range of vessel widths from (less than) a pixel to a maximum width. Since direction estimation may miss the thick vessel, a parameter M for the maximum allowed reconnection distance between the endpoint and a thick vessel is considered. M should be set to half of the maximum vessel width. We accept the thin vessel to be connected to the potential thick vessel if the thick skeletal pixel and the thin endpoint pixel can be connected in this way. If the algorithm is not able to find a thick skeletal pixel for a thin endpoint than it resumes the skeleton extracted from the whole vascular system for that location.

4. Experimental results

For our experimental tests we considered a database of 130 vessel intersections extracted from the 19 binary vascular images of the DRIVE database⁸. The images in the DRIVE database have the resolution of 565x584 pixels with corresponding vessel widths of 1...10 pixels. Considering this dataset, the parameters of our algorithm have been adjusted as follows for optimal performance:

- removing pixels having less than N neighboring pixels:
 $N=4$,
- maximum allowed reconnection distance:
 $M=5$,
- distance from the endpoint for direction estimation:
 $P=6$.

To see the improvement, we have compared the skeleton resulted by our approach with the classical skeleton extracted from the original intersection image without any splitting/reconnection step. For a quantitative comparison, we considered the degree of distortion (*DD*) term defined in section 2.3.

From the 130 intersections, our algorithm generated 47 different skeletons than classical skeletonization. That is, basically in 36% of the cases the segments had sufficiently different widths for a reliably split. Comparing the results with respect to the *DD* term, we got that our algorithm gave better results for 29 intersections. In 14 cases, the detected junction points were different, but the *DD* term remained the same. There were 2 intersections, where our algorithm gave worse results than the classical skeletonization. In these cases, the thin vessels can be considered rather as circular segments than linear ones causing a failure for direction estimation.

Considering the *DD* term for the 130 intersections, our algorithm reduced the total *DD* error from 140 to 72 pixels that means a 48% suppression of distortion for these cases. Our results are also summarized in Table 1.

Splittable intersections (47/130)		
Skeletonization:	Classic	Proposed
Total <i>DD</i> error (pixels):	140.0	72.0
Average <i>DD</i> error (pixels):	3.11	1.6
Distortion decreased to: 51.4286 %		
All intersections (130/130)		
unsuccessful splitting		85
better results with proposed method		29
same results with proposed method		14
worse results with proposed method		2

Table 1. Improvement of the proposed method with respect to classical skeletonization for 130 vessel intersections.

Regarding the experimental tests reported above, the question may arise that in general how many intersections can be found in a human retina with sufficient width differences for a successful splitting. To get an estimation, we considered a wide range retina image with 200 degrees field of view shown in Figure 10a. Then, we extracted manually the vessel system and tested our algorithm with the corresponding parameter settings. Figure 10b depicts all the 29 positions, where the algorithm was able to split the intersecting vessel segments and thus, to reduce the degree of distortion.

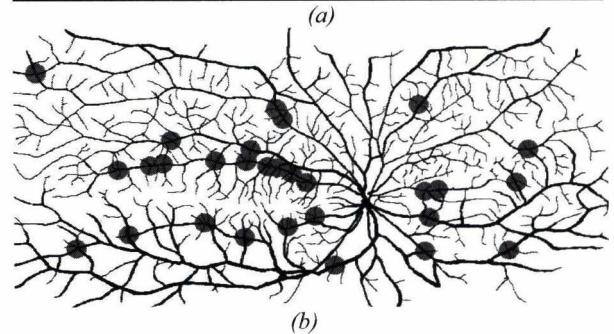
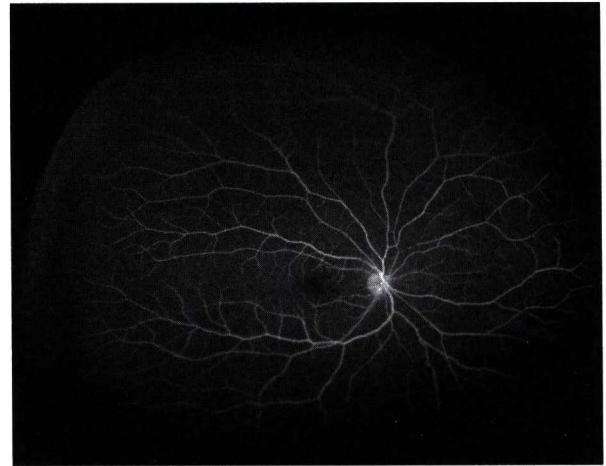


Figure 10: Thick/thin intersections on a wide range retinal image. (a) original image (from www.optos.com), (b) manually segmented vascular system with improved intersections marked by gray discs.

5. Discussion and conclusion

The DRIVE database contains both macula and optic disc centered images. If the retinal image is centered at the macula, then the vessels have higher curvature, while for an image centered at the optic disc, they are more linear. As a consequence, images centered at the macula contain more intersecting curve segments of larger curvature. Thus, the linear direction estimation approach gives more reliable results for images centered to the optic disc. Since we intend to process images of both types, we did not make difference between them and calculated an overall performance of our algorithm in the tests. However, to refine our and other approaches, quadratic or higher order interpolation and direction estimation may lead to improved performance. Moreover, our proposed method should improve the skeleton for fork (Y-shaped) junctions, which issue has not been pursued in this paper.

6. Acknowledgement

This work was supported in part by the Janos Bolyai grant of the Hungarian Academy of Sciences, and by the TECH08-2 project DRSCREEN – Developing a computer based image processing system for diabetic retinopathy screening of the National Office for Research and Technology of Hungary (contract no.: OM-00194/2008, OM-00195/2008, OM-00196/2008).

7. References

- [1] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, New York, 1982.
- [2] A. Hajdu, and I. Pitas. Piecewise linear digital curve representation and compression using graph theory and a line segment alphabet. *IEEE Trans. on Image Processing*, **17**(2):126–133, 2008.
- [3] D. Zhong, and H. Yan. Pattern skeletonization using run-length-wise processing for intersection distortion problem. *Pattern Recognition Letters*, **20**(8):833–846, 1999.
- [4] R. Al Ajlouni. The use of digital pattern recognition techniques for virtual reconstruction of eroded and visually complicated archeological geometric patterns. *ISPRS08*, B5:193–198, 2008.
- [5] M. Niemeijer, J. J. Staal, B. van Ginneken, M. Loog, and M. D. Abramoff. Comparative study of retinal vessel segmentation methods on a new publicly available database. *SPIE Medical Imaging*, 5370:648–656, 2004.
- [6] E. S. Deutsch. Thinning algorithms on rectangular, hexagonal, and triangular arrays. *Communications of the ACM*, **15**(9):827–837, 1972.
- [7] R. Klette, and A. Rosenfeld. *Digital Geometry*. Morgan Kaufmann, San Francisco, 2004.
- [8] J. J. Staal, M. D. Abramoff, M. Niemeijer, M. A. Viergever, and B. van Ginneken. Ridge based vessel segmentation in color images of the retina. *IEEE Trans. on Medical Imaging*, **23**, 501-509, 2004.

Well fitting curve models with few parameters for vascular systems on retinal images

I. Csosz¹, A. Csutak², T. Peto³ and A. Hajdu¹

¹Faculty of Informatics, University of Debrecen, 4010 Debrecen, POB 12, Hungary

²Medical and Health Science Center, University of Debrecen, Nagyerdei Krt. 98, 4032 Debrecen, Hungary

³Moorfields Eye Hospital, 162 City Road, London EC1V 2PD, United Kingdom

Abstract

Diabetic retinopathy is one of the most common causes of blindness. Thus, a growing demand on automatic screening has risen. Among the anatomical parts, thick vessels can be observed most reliably in fundus images. Especially, the localization of the temporal arcade is highly important to detect other anatomical parts. State-of-art detectors locate the arcade by fitting semi-ellipses or parabolas. However, the selection of these models is rather ad hoc. We perform a thorough analysis to find curve models that fits optimally. We use non-linear regression analysis with corresponding error measurement. Several models with low numbers of parameters are checked to support fast matching. Comparative results with the current models are also shown.

Index Terms— curve fitting, blood vessels, biomedical imaging, image matching

1. Introduction

Diabetic retinopathy (DR) is one of the most common causes of blindness in the developed countries¹. Screening of retinopathy is essential to prevent blindness of patients with diabetes. The manual screening methods have both high financial cost and human resource needs. Nowadays, several approaches have been considered to build automatic computer-based screening programmes^{2,3}. One of the most reliably detectable anatomical parts – even for low quality images – is the vascular system, because there are no similar lesions on the retina. It is also obvious that thicker vessels can be detected more easily. Thickest vessels belong to the temporal arcade, which is highlighted in Figure 1a. The proper localization of the temporal arcade has several advantages. For example, the fundus image can be classified, whether it has been captured from the left or right eye of the patient. Moreover, the optic disc and macula (see Figure 1b) can be detected more reliably regarding their relative position to the temporal arcade^{4,5}.

Our group currently develops an automatic screening system for diabetic retinopathy, which also considers the detection of temporal arcade for the proper navigability of the image. Prior approaches for temporal arcade detection are based on fitting semi-ellipse⁴ or parabola⁵ models to the vascular system. Though the shape of the temporal arcade intuitively suggests these models, quantitative analyses have not been performed to decide which model should be considered for optimal fitting. Thus, these choices are rather ad hoc. To resolve this issue, we have executed a thorough theoretical investigation. That is, our aim was to find parametric curve models that approximate the shape of the temporal arcade optimally regarding a proper error term. We use non-linear regression tests on temporal arcades with Akaike Information Criterion (AIC) for error measurement to find the best fitting models. Low parameter number of the fitted model is highly welcome for fast matching algorithms. Thus, we have focused on curve models having three parameters, similarly to the parabola⁵.

The rest of the paper is organized as the follows. In section 2, we present the theoretical background

that is used for fitting the curve models. Section 3 contains the list of the curve models found, ranked by their fitting performance measured by AIC. A minor improvement is applied to the best fitting models in section 4 to raise fitting accuracy. Finally, some conclusions are drawn in section 5.

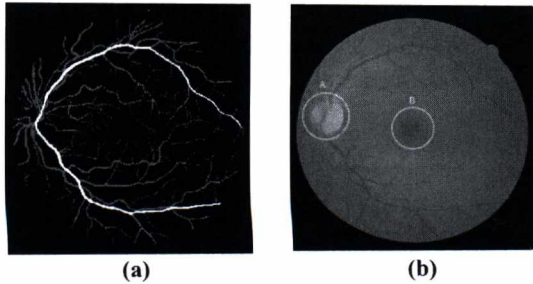


Figure 1: Main anatomical parts of the retina: (a) the vascular system with the temporal arcade highlighted, (b) the optic disc (A) and the macula (B).

2. Curve fitting approach

Finding analytically the best fitting curve models with low numbers of parameters is a very challenging problem. However, finding models of approximating curves with regression analysis has a wide literature and strong support from both theory and scientific software packages. Especially, strongest support is available for 1D, when the input data are supposed to be “function-like” (that is, basically one vertical coordinate for any horizontal coordinate). In this case, a real valued parameterized functional model can be used for fitting. In our case, some simple preprocessing steps can make the temporal arcade be “function-like” to be able to take advantage of the strong scientific background.

2.1. Preprocessing

To reach a representation of the temporal arcade that is suitable for nonlinear regression by functional models, the original temporal arcade (see Figure 2a) has to undergo some simple transformations. Namely:

- it is rotated by 90 degrees to CW or CCW direction (based on, whether it belongs to a left or right eye image), see Figure 2b,
- cut at half of its height, see Figure 2c.

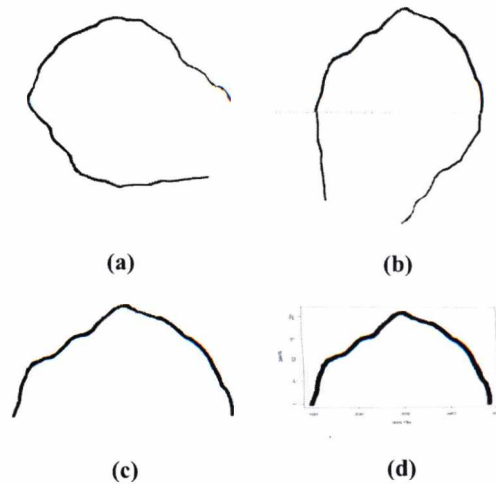


Figure 2: Preparing of the temporal arcade for nonlinear regression: (a) original temporal arcade, (b) rotation by 90 degree with cropping line marked, (c) result of cropping, (d) the temporal arcade considered for non-linear regression.

2.2. Non-linear regression

Several scientific software packages (SPSS, SAS, etc.) support the definition of any parameterized models in terms of a combination of basic mathematical functions. However, it is very challenging to set up an intuitive model that can be expected to fit the shape of the temporal arcade. To overcome this difficulty, these software packages offer built-in models to perform curve fitting. Especially, the DataFit 9.0 software offers nearly 300 built-in models for this aim⁶. Its range covers (modified) power, (modified) exponential, rectangular hyperbola, reciprocal, logarithm, geometric, hyperbolic and trigonometrics. These models have been found useful in scientific, statistical and engineering applications. Thus, in our investigations we also relied on these types to find primary models that can be improved further on. Non-linear regression has been performed in the open source statistical programming language R⁷ which also supports the calculation of the currently recommended error term AIC for regression. Since R also has the ability to define custom models, we could perform a non-linear regression using nearly 300 curve models borrowed from DataFit. For an

impression, see Figure 3 for the best fitting curve model found for a temporal arcade in this way.

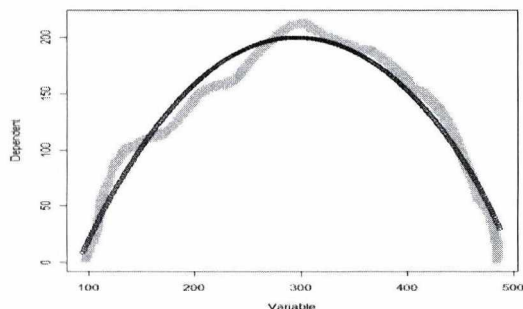


Figure 3: Result of non-linear regression on a temporal arcade. The temporal arcade is depicted in gray, while the best fitting three parameters curve model is in black.

2.3. Measuring the fitting performance

Following the state-of-the-art statistical recommendations, we considered the Akaike Information Criterion (AIC)⁸ to find a ranking of the models considered for fitting. AIC is a measure of the goodness of fit of an estimated statistical model. The formal calculation of AIC is given by

$$AIC = 2 * k + n * [\log(2 * \pi * R / n) + 1]$$

where n is the number of sample points, k is the number of parameters in the fitted model, and R is the residual sum of squares. That is, smaller AIC value indicates better fit. Also note that AIC can be used to punish the number of parameters considered in the model. In our experiments, we considered the default $k = 2$ punishment of the software package R. In general, the number of model parameters should be kept low for matching algorithms. As a typical example, we can mention the popular Hough transformation⁹, which can be applied to find the best matching of any analytically known model. However, the space and time complexity of the Hough transformation can be approximated by $O(s^p)$ and $O(s^{p-1}m)$ respectively, where m is the number of points, p is the number of parameters and s is the number of samples along one Hough dimension. That is, low number of parameters can be a crucial issue regarding computational performance.

3. Finding the best fitting models

In our experiments, we considered manually segmented temporal arcades to perform the regression analysis. Moreover, since in an automatic screening system the computational performance is also a crucial issue, we focused on curve models with three parameters. However, in our comparative results we also exhibit some four parameters models to be able to see the performance of the semi-ellipse fitting method⁴, as well.

3.1. Test data set

To be able to compare the fitting performance of the curve models, we used the publicly available DRIVE database¹⁰ which consists of 40 fundus images together with their manually segmented vascular systems. From the 40 images, there are 35 which are macula centered and are suitable for temporal arcade detection. All images have the same resolution of 565 x 584 pixels. Since the manually segmented vascular systems in the DRIVE database contain all the vessels, we manually extracted the temporal arcades from them. In this way we got a test database containing 35 manually segmented temporal arcades. Some elements of the database are shown in Figure 4.



Figure 4: Manually selected temporal arcades from the data set considered for experimental tests.

3.2. Ranking the fitted models

To be able to obtain an overall ranking of the approximately 300 curve models, we calculated the average of the AIC values of each of the models measured for the 35 elements of the data set, and ranked them accordingly. Table 1 summarizes the top 10 models with three parameters (a, b, c). Moreover, the four parameters semi-ellipse and third-order polynomial models are also included for comparison. As the semi-ellipse model is not included in the DataFit software, we modeled it as:

$$(d^2 * (1 - (x - a)^2 / c^2) + b)^{0.5}.$$

That is, the ellipse is centered at (a,b) and have 2c for its horizontal, and 2d for its vertical diameter, respectively.

Models with four parameters		
Rank	Fitted model	Average AIC
1.	<u>Third- order polynomial</u> $a*x^3 + b*x^2 + c*x + d$	33463.875
2.	<u>Semi-ellipse</u> $(d^2 * (1 - (x - a)^2 / c^2) + b)^{0.5}$	34226.011
Models with three parameters		
3.	$a*x^3 + b*log(x)^2 + c$	34390.211
4.	$a*x^{2.5} + b*x^{0.5}*log(x) + c$	34407.671
5.	$a*x^{2.5} + b*x^{0.5} + c$	34428.254
6.	$a*x^2*log(x) + b*x/log(x) + c$	34438.833
7.	$a*x^3 + b*x^{0.5} + c$	34443.034
8.	<u>Parabola</u> $a*x^2 + b*x + c$	34447.561
9.	$a*x + b*x^2*log(x) + c$	34456.069
10.	$a*x^{2.5} + b*x/log(x) + c$	34485.314
11.	$a*x^2 + b*x/log(x) + c$	34490.407
12.	$a*x*log(x) + b*x^2 + c$	34495.301

Table 1: Ranking of the best fitting curve models with three and four parameters.

From the table we can deduct the followings:

- better fitting models with three parameters can be found than the parabola,
- the four parameters semi-ellipse has slightly better fit than the three parameters models,
- four parameters models can be found easily with much better fit than the semi-ellipse.

4. Refining fitted models

The basic models presented in Table 1 can be further improved in a simple way. To do so, let us consider the following with three parameters generalized model:

$$a * f(x, u_1, u_2, ...) + b * g(x, v_1, v_2, ...) + c.$$

The idea for refinement is to let the scalar parameters u_1, u_2, \dots , and v_1, v_2, \dots change slightly. For example, if the original model (best ranked one) is:

$$a * x^3 + b * log(x)^2 + c,$$

then we consider its generalization as:

$$a * x^u + b * log(x)^v + c$$

and try to find a better fitting curve with letting $u \in [2.5, 3.5]$, and $v \in [1.5, 2.5]$, respectively. Intuitively, we let the basic models to change a bit to gain better fitting performance. The improved versions of the best fitting first six models with three parameters are shown in Table 2.

Refined models with three parameters		
Rank	Fitted model	Average AIC
1.	$a*x^{3.28} + b*log(x)^{1.71} + c$	34369.481
2.	$a*x^{3.27} + b*x^{0.15} + c$	34369.712
3.	$a*x^3 + b*x^{0.3} + c$	34374.218
4.	$a*x^3 + b*x^{0.2}*log(x) + c$	34374.462
5.	$a*x^{2.5}*log(x) + b*x^{0.62}/log(x) + c$	34394.397
6.	<u>Refined parabola</u> $a*x^{2.5} + b*x^{0.62} + c$	34404.796

Table 2: Ranking of the best fitting curve models with three parameters after refining them.

From the table we can deduct the followings:

- after the proposed refinement better fitting can be reached starting from the basic models,
- the rank of the parabola model remained the same,
- some of the models changed their rank regarding Table 1,
- the first four models of Table 2. can be recommended primarily for matching purposes.

5. Conclusion

In this paper, we examined which curve models with low number of parameters can be fitted well to temporal arcades in retinal images. As it can be checked in Table 1 and 2, we have determined three parameters models with better fit on temporal arcades than the parabola. Moreover, these three parameters model – especially, after a refinement – have comparable performance to that of the four parameters semi-ellipse. However, from Table 1 we can see that finding well fitting four parameters models is a promising approach, since even a simple model (polynomial) may improve fitting accuracy in a large extent.

Though some improvements have already been found by our approach, several issues raise which should be further investigated. For example, we could

consider vector regression¹¹ directly for the 2D temporal arcade instead of originating it to 1D. However, computational drawbacks might be expected if the coordinate functions of the temporal arcades are investigated separately because of the growing number of model parameters. A promising extension can be to check, whether temporal arcades can be organized into clusters to use different models for the clusters, if so. The models found by non-linear regression can be directly used in matching scenarios (e.g. by Hough transformation) or we can compose binary templates to be detected based on them. Finally, four parameters models could be checked in a similar way. However, much less scientific support is available currently for this aim.

Acknowledgement

This work was supported in part by the Janos Bolyai grant of the Hungarian Academy of Sciences, and by the TECH08-2 project DRSCREEN- Developing a computer based image processing system for diabetic retinopathy screening of the National Office for Research and Technology of Hungary (contract no.: OM-00194/2008, OM-00195/2008, OM-00196/2008).

References

1. American Diabetes Association, "National Diabetes Fact Sheet", 2007.
2. Abramoff, M. Niemeijer, M.S.A. Suttorp-Schulten, M. A. Viergever, S. R. Russel, and B. van Ginneken, "Evaluation of a system for automatic detection of diabetic retinopathy from color fundus photographs in a large population of patients with diabetes," *Diabetes Care*, vol. 31, pp. 193-198, February 2008.
3. O. Hejlesen, B. Ege, K-H. Englemeier, S. Aldington, L. McCanna, and T. Bek, "Tosca-imaging developing internet based image processing software for screening and diagnosis of diabetic retinopathy," *MEDINFO*, pp. 222-226, 2004.
4. A.D. Fleming, K.A. Goatman, S. Philip, J.A. Olson, and P.F. Sharp, "Automatic detection of retinal anatomy to assist diabetic retinopathy screening," *Phys. Med. Biol.* vol. 52, pp. 331-345, 2007.
5. S. Ravishankar, A. Jain, and A. Mittal, "Automated feature extraction for early detection of diabetic retinopathy in fundus images," *IEEE Conference on Computer Vision and Pattern Recognition*, pp. 210-217, Miami Beach, FL, USA, 2009.
6. Oakdale Engineering, "DataFit 9.0," 2009. URL: <http://www.oakdaleengr.com/>.
7. R Development Core Team, "R: A Language and Environment for Statistical Computing," 2009. URL: <http://www.R-project.org>.
8. H. Akaike, "A new look at the statistical model identification," *IEEE Trans. on Automatic Control*, vol. 19, pp. 716-723, 1974.
9. R.O. Duda, and P.E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," *Comm. ACM*, vol. 15, pp. 11-15, 1972.
10. J.J. Staal, M.D. Abramoff, M. Niemeijer, M.A. Viergever, and B. van Ginneken, "Ridge based vessel segmentation in color images of the retina," *IEEE Trans. on Medical Imaging*, vol. 23, pp. 501-509, 2004.
11. L. Baldassarre, A. Barla, B. Giancesin, and M. Marinelli, "Vector valued regression for iron overload estimation," 19th International Conference on Pattern Recognition, Tampa, FL, USA, pp. 1-4, 2008.

Atlas-based abdomen segmentation

A. Kriston, Gy. Bekes, L. Rusko, M. Fidrich

Abstract

This paper presents an anatomical atlas based framework for automatic abdominal segmentations. The framework consists of inter-patient registration, atlas construction, automated seed region generation and active contour based segmentation. The presented framework was evaluated on a set of 12 CT exams, which shows that it can be used for automated segmentation of abdominal organs: mediastinum, liver, spleen, left and right kidney.

1. Introduction

In radiation treatment and surgery planning clinicians need to delineate the contour of anatomical structures on a large number of 2D slices. There are different semi-automatic methods [1][2], which facilitate the contouring and decrease the processing time. An automatic segmentation method could further decrease the processing time and makes it possible to establish new clinical work-flow, wherein the algorithms are executed in the background (on a server) before the user starts to work with the images. The automation of contouring also eliminates the problems related to inter- and intra-operator variability.

In this paper we focus on the automated segmentation of organs in the abdomen and the chest on native CT images. In the abdominal region the targeted organs are the liver, the kidneys, the spleen and the spinal cord. In the chest we segment the left and right lung and the mediastinum. Mediastinum is a thoracic area, which contains the heart, the vessels (aorta, vena cava, pulmonary arteries, and pulmonary veins), lymph nodes, the bronchus and the esophagus and it is surrounded by the diaphragm, the lung lobes, the spine, and the sternum. This area plays a very important role in oncology (lung cancer, Hodgkin and Non-Hodgkin Lymphoma, etc.).

2. Previous work

CT is the most widespread modality in oncology. We have earlier developed semi-automatic methods for liver, spleen and kidney segmentation [1], which have been successfully used in radiotherapy planning. The common feature of these methods is that they require

user initialization (in form of a trace or some seed points inside the organ) and all of them is based on active surface method [3]. The user-given input is used to compute intensity statistics and generate the initial deformable surface, which is inflated according to organ specific parameters.

Automating the above-mentioned contouring methods is challenging because clinical expertise shall be incorporated in the algorithm to identify the location of the target organs. In case of mediastinum the segmentation of the surrounding organs is also necessary to determine the precise contour of the region of interest. In order to solve these problems we constructed a probability atlas that is used for automated initialization of the existing methods as well as developing new methods.

3. Detailed description

3.1. Atlas construction

Our goal was to build a probabilistic atlas that is based on several manually segmented CT images. The atlas involves a probability map for each organ. These maps are placed in a common coordinate system, so the relative locations of the organs are also represented.

13 full-body exams were selected for the atlas construction and according to the general clinical segmentation needs 13 anatomical structures (femur heads, bladder, pelvic, liver, left and right kidney, spleen, spine, left and right lung, mediastinum, ribs and shoulders) were manually contoured on each image.

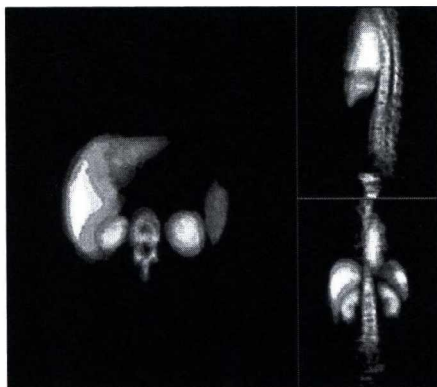


Figure 1 The probabilistic atlas. Different colors represent different organs.

These scans were registered to a reference image in order to build the atlas. The reference image of the atlas was chosen by the following steps:

- An automatic algorithm registered all possible image pairs
- The results were grouped by moving image (the image which is transformed during the registration)
- The average overlap of the liver, spleen, kidneys and mediastinum was computed in each group
- The moving image belonging to the group with highest average overlap was selected as the reference

All segmented images were registered to this reference and the segmented structures were added to the atlas.

3.2. Registration

Our registration method consists of 2 main steps. The first step is whole-body registration based on the skeleton of the patients, which is done in the following way:

- Pre-processing steps remove the heads, arms, high intensity artifacts and contrast agents from the two whole-body input CTs if necessary.
- Bone profiles are calculated for both CTs: for each slice along the z-axis the sum of the distance of bone voxels from the center of the slice is computed.
- An initial scaling and translation belonging to the z-axis is computed by registering the profile of the moving image to the profile of the fixed image.

The second step focuses on a selected body-part. In this case, an abdominal region is automatically determined and clipped from the reference image and based on the scaling and translation results of the first step, the corresponding region is clipped from the fixed image.

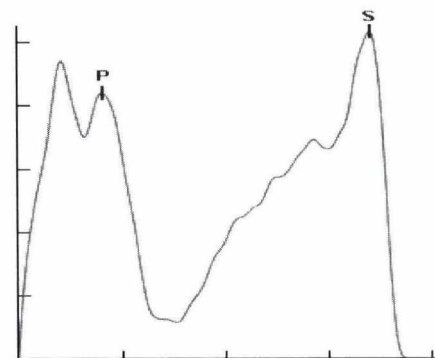


Figure 2 The bone profile of a typical patient, horizontal axis represents slice number, and vertical axis represents the value of bone profile. P and S denote the top of the pelvis and the shoulder area, respectively.

Performing a 3D similarity registration is very time-consuming, therefore we apply a faster approach. Our algorithm generates 2D raysum projections (axial and coronal) from the clipped regions, which are used to perform 2 separate anisotropic similarity transformation based 2D registrations. The axial projections are used to calculate the translation and scaling in x-y direction, and the registration of the coronal projections results the z translation and scaling. The values of the 3D transform matrix are defined by the corresponding values of the 2D transformation matrices.

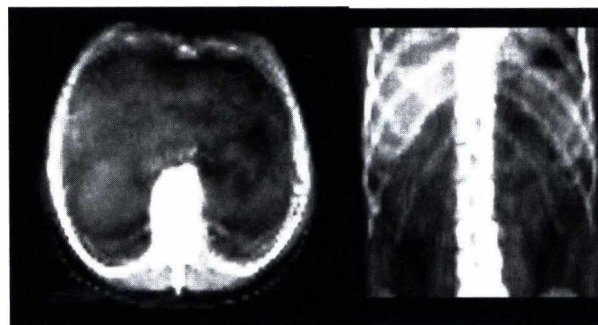


Figure 3 Axial and coronal raysum projections.

3.3. Segmentation

3.3.1. Automating existing methods

The automatic segmentation starts with a registration, where the reference patient of the atlas is registered to the currently examined patient using the registration method described in the previous section. The resulted transformation is applied to the atlas.

In order to automate a semi-automatic method, a connected volume (initial region) is determined within the organ. To determine this region, we generate a volume from voxels, which fulfill the following two criteria:

- a) the probability value in the atlas exceeds a predefined probability, and
- b) the intensity on the CT image is in a predefined range.

To prevent the appearance of bone voxels (e.g. ribs) in the vicinity of this volume (requirement of the organ segmentation), voxels having high intensity (>150 HU) neighbors in the CT image are removed (the radius of the neighborhood is 10mm). The resulted volume may be fragmented, that is why only the largest connected volume is used as initial region.

The segmentation methods were modified to generate their initial deformable surface from this region. Originally, the algorithms calculated the statistics (modus, average, scatter, inflation force) from the environment of the user defined input curve and the initial surface was constructed from icosahedrons along the curve. After the modification, the methods compute statistics from the initial region and generate the initial surface from that.

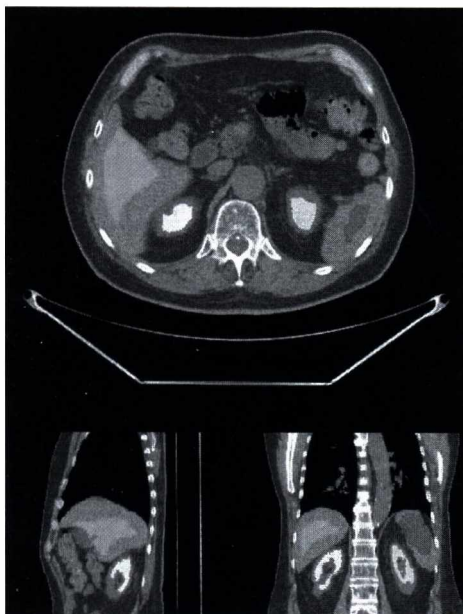


Figure 4 Generated seed regions

3.3.2. Mediastinum segmentation

Besides the automation of existing methods a new method was also developed for automatic mediastinum segmentation. This anatomical area involves many organs and is surrounded by abdominal and thoracic organs: liver, spleen, spine, ribs and lung lobes. The method incorporates the results of the automatic segmentations (liver, spleen, lung) to generate the spatial boundaries of this area, which is followed by the precise segmentation of the mediastinum.

The top and bottom boundaries of the area are determined by the spleen and the lungs. Bone structures are detected by intensity and probability values. Thereafter the consecutive ribs are connected with straight lines on the axial slices, which is followed by a convex hull algorithm to close the mediastinum area in the direction of x and y axes. Finally, the automatically segmented organs (lungs, liver and spleen) are removed from this region.

In order to determine the precise contour of the mediastinum a 3D region growing is started from the weight center of the voxels, where the probability of the mediastinum region is greater than 90%. If this point belongs to another organ (because one of the segmented organs is over-segmented into the mediastinum) the result of the automatic segmentation is corrected in the following way. Each voxel is erased from the organ, wherein the probability of the mediastinum is higher than the probability of the given organ.

4. Results

By now, three semi-automatic methods (liver, spleen and the kidneys) were automated, and a new method for mediastinum segmentation was developed. The lung segmentation was originally automated, so we could use it without modification.

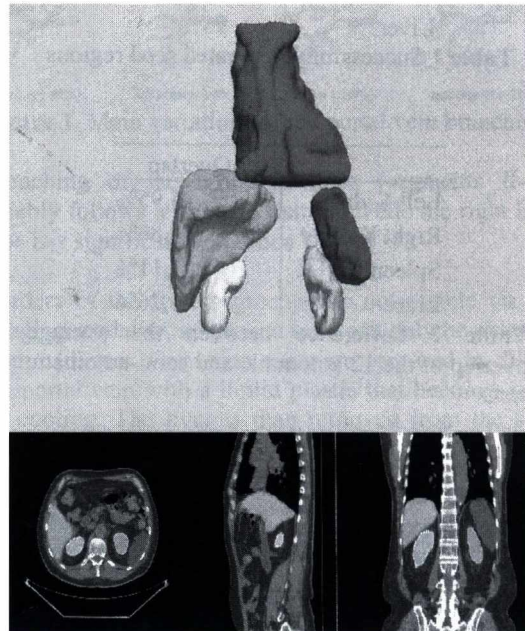


Figure 5 Results of the automatic segmentations for a typical patient's data. The image shows the mediastinum (blue), the liver (green), the spleen (red) and the two kidneys (yellow and purple).

To measure the robustness of the atlas registration, seed regions were generated for liver, spleen and kidneys in 24 whole-body CT scans. 96% of the seed regions for the liver and spleen, and 88% of the seed regions for the kidneys were correct for segmentation initialization (Table 1). The most difficult problem was the automation of the kidney segmentation due to the high inter-patient variability of its location and size.

To compare the accuracy of the semi-automatic and the automated methods, the overlap between the real and the generated contour was measured for 12 test cases using both the semi-automatic and the automatic methods. As seen in Table 2, the difference between the results of the semi-automatic and automated methods is not significant. The liver and spleen automation produced better results, but the results of automated kidney segmentation show decrease in accuracy.

The mediastinum area consists of different organs, that is why it is hard to determine the correct boundary. Despite of this difficulty, the automatic mediastinum segmentation produced good results, average 87% true positive and 7% false positive volume fractions.

	Correct seed regions
Left Kidney	88%
Right Kidney	88%
Spleen	96%
Liver	96%

Table 1 Successfully generated seed regions

	Δ Overlap
Left Kidney	-5,95%
Right Kidney	-3,86%
Spleen	1,11%
Liver	3,26%

Table 2 Difference between the average overlap of the 12 automatic and semi-automatic tests

5. Conclusion

The primary goal of the project was to automate the existing semi-automatic methods without sacrificing the segmentation quality.

We developed an automatic segmentation framework based on probabilistic atlas. This framework showed great potential in automation of existing semi-automatic methods and it can be the basis of developing new methods. The built-in clinical

knowledge allows fast and accurate automatic organ segmentation, which is able to run in the background, saving significant time for the clinicians. However, there are some points where this framework can be improved (e.g. more accurate registration; using the atlas during the segmentation not only for initialization). This framework was tested for native CT scans, but in the future it can be applied to other modalities (MR).

As major benefits of automation, the eliminated operator-dependency and the improved repeatability are the greatest achievements, furthermore the precision of the segmentation methods was preserved.

6. References

- [1] György Bekes, László G. Nyúl, Eörs Máté, Attila Kuba, Márta Fidrich (2007) 3D segmentation of liver, kidneys and spleen from CT images. *International Journal of Computer Assisted Radiology and Surgery*, Springer Verlag, vol. 2, S45-S47.
- [2] György Bekes, Imre Papp, Márta Fidrich, Attila Tanács (2008) Semi-automatic 3D segmentation toolkit and its application to pelvic organs. *Proceedings of CARS*
- [3] McInerney T, Terzopoulos D (1999) Topology adaptive deformable surfaces for medical image volume segmentation. *IEEE Trans Med Imag* 18(10): 840-850
- [4] László G. Nyúl, Judit Kanyó, Eörs Máté, Géza Makay, Emese Balogh, Márta Fidrich, Attila Kuba. Method for Automatically Segmenting the Spinal Cord and Canal from 3D CT Images. In A. Gagalowicz and W. Philips, editors, *Proceedings of International Conference on Computer Analysis of Images and Patterns*, volume 3691 of *Lecture Notes in Computer Science*, Heidelberg, pages 456-463, September 2005. Springer Verlag.

Semi-automatic framework for anatomical liver segment separation

T. Ungi, A. Kriston, T. Blaskovics, M. Fidrich

Abstract

This paper presents an automatic framework for portal vein segmentation and liver segment separation. The framework consists of hepatic vessel segmentation, portal and hepatic vein separation, automated labeling of main vein branches and anatomical segment separation modules. Various methods are presented and discussed for the different modules. The preliminary evaluation involving some representative cases shows the functionality of the proposed framework.

1 Introduction

The liver can be divided into 8 functionally independent segments. For liver surgical planning, structure and morphology of the hepatic vessels are of major interest.¹ The clinical applications of the separation of liver segments are:²

- Liver transplants, when a healthy donor gives a part of his or her liver to another person. Volume of the resected part has to be determined to predict postoperative liver function. The resection is best to be made respecting liver segment boundaries.
- Oncologic resection, and its extent of resection depend on the location of tumors and the spatial relations between tumors, hepatic vessels and segments.

In the center of each segment there is a branch of the portal vein, hepatic artery and bile duct. In the periphery of each segment there is vascular outflow through the hepatic veins.

Right hepatic vein divides the right lobe into anterior and posterior segments.

Middle hepatic vein divides the liver into right and left lobes (or right and left hemiliver). This plane runs from the inferior vena cava to the gallbladder fossa. Left hepatic vein divides the left lobe into a medial and lateral part. Portal vein divides the liver into upper and lower segments. The left and right portal veins branch superiorly and inferiorly to project into the center of each segment.³

Branching of the portal vein respects liver segment boundaries. The 8 greatest branches represent the 8 functional segments of the liver. Conventional main portal branching is present in about 65% of the patients.⁴

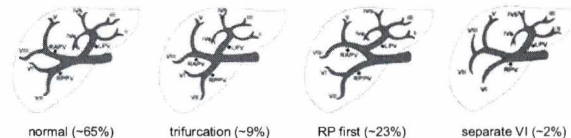


Figure 1. Main variations of the portal vein branching⁵

Branching of the left liver lobe (segments II-IV) reliably follows a standard pattern, while the right liver lobe has significant variations.⁶

Borders of the liver segments are not visible on any imaging modality, so ground truth can only be obtained from cadavers. Corrosion casts are prepared by filling the portal vein with a liquid plastic that becomes solid by cooling. The liver is then removed from the solid plastic that leaves the detailed branching system of the portal vein.

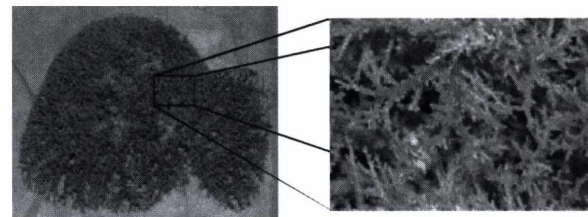


Figure 2. A portal corrosion cast⁷

2 Previous Work

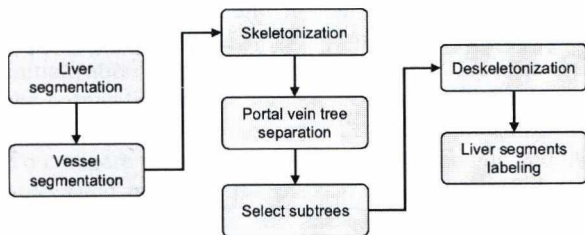


Figure 3. The main modules of a published solution, which was followed by our earlier work²

2.1 Vessel segmentation

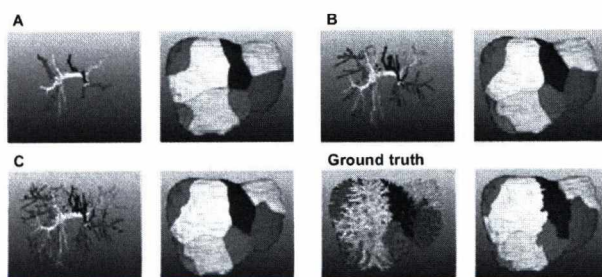
Intensity based region growing from seed points can segment portal veins until the third branching level. The problems with this simple method are

- 1) oversegmentation in the hepatic vein,
- 2) undersegmentation in case of smaller branches.

Undersegmentation can be reduced by a hybrid method of combining morphological closing, distance analysis and thresholding.⁸

A Hessian eigenvalue based line filtering could greatly enhance the accuracy of vessel segmentation.⁸ Reducing the region of enhancement to the region of segmented liver could reduce computation time under an acceptable limit.

Branching points do not have a characteristic tubular shape, therefore region growing on a vesselness enhanced image can stop at branching points. These points can be detected by corner detection methods, and handles separately to connect close vessel areas.¹⁰



	A	B	C
Overlap with Ground Truth	79 %	90 %	93.4 %

Figure 4. Impact of vessel segmentation accuracy (A: only the main branches, B: with secondary branches, C: third branching level)²

2.2 Impact of vessel segmentation

Portal vein segmentation is the input of segment separation. The accuracy of portal vessel segmentation has a high impact on the accuracy of segment separation.

If the vessel segmentation method can only detect the 8 main portal branches, we can expect a 25% volumetric error in the segment separation results, which is unacceptable. At least the secondary branches should be accurately segmented to reduce the error under 10%.

2.3 Portal vein tree separation

Usually two different vessel systems of the liver are enhanced with contrast during the scan, which results in an oversegmentation of the portal system into the hepatic vein system.

Leakage of the segmented portal region into the hepatic vein can be characterized with creation of loops in the topology of the region (note that loops can occur also between portal branches), increase in thickness of the branches in the blood flow direction and obtuse angles in branch junctions.⁸

Identification of these features enables us to automatically remove nearly all non portal vessel segments.

Another method, reported to work automatically in most of the cases² has several parameters, some of which has to be adjusted manually. The tree separation works on graph representation of the vessels (output of skeletonization), because it enables an easier access to the geometry of branches and structural information.

2.4 Skeleton to graph conversion

Palagyi et al. successfully determined the topological graphs of airway skeletons.¹¹ He used the breadth-first search algorithm to go over the determined skeletal branching points. This method constructs a graph of tree topology for all skeletons. However, liver vessels contain circles, even after an accurate segmentation, due to the coexistence of more vessel trees, and the partial volume effect. The breadth-first search should be modified for our purpose.

2.5 Subtree selection

Selle et al. identifies the 8 most voluminous subtrees in the segmented portal system. Thus, their system will not be able to correctly label a patient's portal vein after a segmentectomy, or a patient's portal vein with some topological exception.

Soler et al. use a similar method, but classification of portal branches is guided by a manually segmented liver registered to the actual case. Two portal sub-branches are merged only if they fall into the same

label on the registered template. This method is more reasonable and advanced than the previous.

2.6 Conclusions of literature review

Developments of accurate and robust vessel segmentation and liver segmentation methods are prerequisites for liver segment separation. Evaluation of new methods with corrosion casts, or during liver transplantation surgery is expensive. It is reasonable to use methods published by others that are already tested and evaluated.

3 Our proposal in detail

3.1 Overview

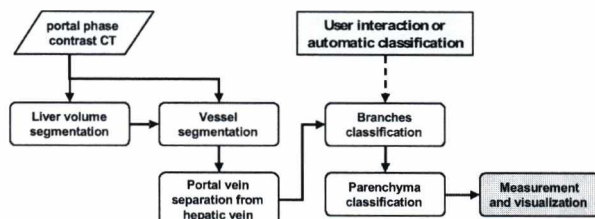


Figure 5. Workflow of our method

As inputs, a portal phase contrast enhanced CT and a segmented liver volume are needed. The segmentation can be done automatically or manually, depends on the available resources.

Vessel segmentation classifies the input volume into vessel and background volumes. Then separates the portal vein system from the segmented vessels. Branches classification labels portal vein branches as they belong to different anatomical segments, this can be classified manually, or automatically. The possibility of manual editing after an automatic classification is still recommended.

Parenchyma classification labels the whole liver volume by mapping each volume element to an identified segmental branch of the portal vein tree.

3.2 Detailed design

3.2.1 Determine parenchyma intensity

Automatic liver parenchyma intensity assessment is based on the anatomical feature, that the liver is the only organ that is on the right side of the abdomen. Therefore, the intensity histogram of the right image side exceeds the histogram of the left image side only at liver parenchyma intensities.

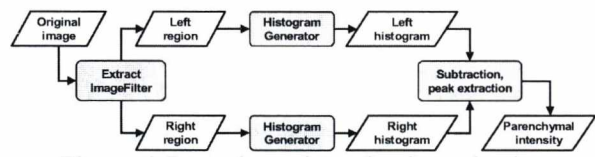


Figure 6. Parenchyma intensity determination

This method gives the true intensity of liver parenchyma on 100% of our test images automatically. The continuous nature of the histogram enables the resulting subtracted histogram to be rescaled between values 0.0 and 1.0. The rescaled histogram can be used as an intensity transfer function that maps the original image to a probability map of belonging to liver parenchyma. This is illustrated in the figure below.

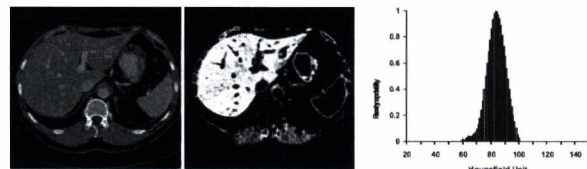


Figure 7. Original CT image slice on the left, parenchyma weighted image in the center, weighted by the intensity transfer function on the right.

3.2.2 Determine vessel intensity

Vessel intensity window determination is a more difficult task, and can be approached by several methods.

Manual selection of regions in the parenchyma and root area of portal tree, in Gaussian filtered ($\sigma=2.0$ mm) images results in the following training data.

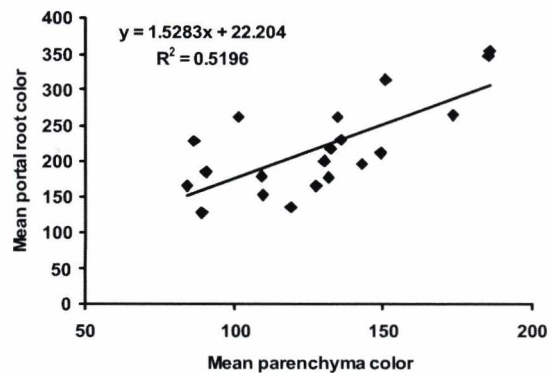


Figure 8. Correlation of liver parenchyma intensity and portal vein root area intensity.

Linear regression leads to the following empirical formula:

$$VesselCenter = ParenchymaCenter * 1.53 + 22.2$$

Where *ParenchymaCenter* is the peak of parenchyma histogram, *VesselCenter* is the estimated peak intensity of the vessel histogram.

It has to be noted that the intensity of smaller vessel branches are much closer to parenchyma intensity than the root area of vessels.

Based on inspection of test image data, 2.5% of the segmented liver volume belongs to vessels. This hypothesis implies that the 97.5 percentile of the histogram of the image covered by segmented liver mask gives a lower threshold for vessels.

This method works well in about half of the test data, but gives too high threshold in case of dilated hepatic veins, and in case of hyperdense tumors in the liver.

3.2.3 Find vessel seed point

Hyperdense tumors may have elongated shapes. Therefore, neither maximal intensity point, nor a line-sensitive shape filter can mark a good seed point for vessels in the liver. An iterative search method is used as shown in the figure below.

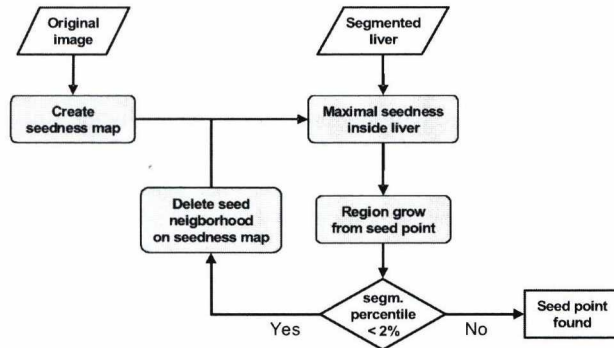


Figure 9. Flowchart of vessel seed point detection.

To create a vessel seed point map and later to enhance the input image, we used the Hessian matrix that describes the local curvature of a function of many variables. If the real-valued function is the following:

$$f(x_1, x_2, \dots, x_n)$$

and if all second partial derivatives of f exist, then the Hessian matrix of f is the matrix:

$$H(f)_{ij}(x) = D_i D_j f(x)$$

where $x = (x_1, x_2, \dots, x_n)$ and D_i is the differentiation operator with respect to the i^{th} argument.

The seedness map is created by Hessian3DToVesselness filter of ITK with 8mm sigma.

Region grow is performed with a lower threshold that is the average of parenchymal intensity, and portal root intensity (by linear regression model). Segmentation is iterated until the segmented volume reaches at least 2% of the liver volume.

3.2.4 Vesselness enhancement

A line filter provides a vesselness measure for tubular objects from the Hessian matrix of image points.¹²

Merging vesselness values and the original image is currently performed by adding vesselness values to the original image intensity.

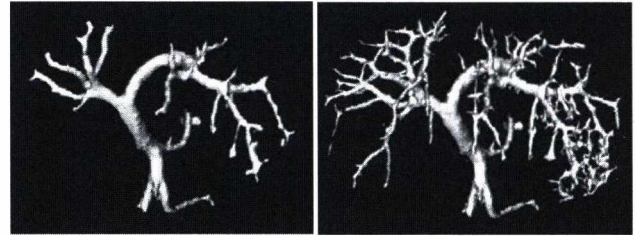


Figure 10. Hessian eigenvalue based vesselness enhancement (right side), improved the result of simple region growing (left side).

3.2.5 Vessel intensity threshold refinement

Selle et al. suggested an optimal threshold determination method for vessel segmentation. The segmented volume increases as the threshold decreases. The increase slope changes when the optimal threshold is passed, because many voxels belonging to the liver tissue are collected. This is shown on the following diagrams.

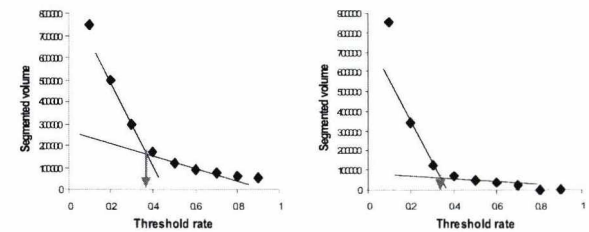


Figure 11. Segmented vessel volume as a function of threshold value. Thresholds are chosen between liver parenchyma intensity and estimated portal vein root intensity.

Threshold rate of slope change is determined by an empirical rule: the intersection of lines determined by points at threshold rates 0.2, 0.3 and 0.5, 0.6.

3.2.6 Skeletonization

We prepare a skeleton of the segmented vessels, and label all skeletal points with its distance from the surface of segmented vessels. Cavities in the segmented vessels cause awkward structures in the

skeleton, not compatible with a tree topology, therefore they have to be filled before the thinning algorithm is applied.

A decision tree based skeletonization method is used.¹³ The algorithm uses 26-neighborhood connectivity for the object and 6-neighborhood connectivity for the background and guarantees unchanged connectedness of the objects as well as not to create holes or cavities in the object.

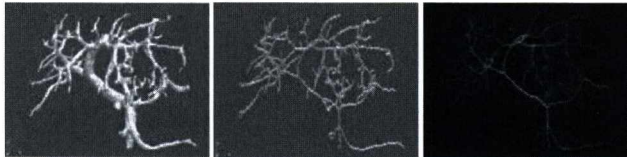


Figure 12. Surface segmented vessels (left), its skeleton (middle) represented by small yellow spheres, and distance labeled skeleton (right).

3.2.7 Skeleton to graph conversion

Portal vein root in the skeleton is found as the point with maximal thickness, and vertical (Z) coordinate is lower than that of the liver volume mass center, has exactly 2 neighboring skeletal points, one of the neighboring points, as a root, have branches that all lead to the liver volume.

Although skeleton to graph conversion has an own literature, we use a simple algorithm to classify skeleton points into branches (edges) of the graph. Points with 2 neighbors are stored by edges that they belong to, all other points are converted to one, and only one, vertex.

Requirements for the skeleton to graph conversion:

- The output graph should be directed.
- An edge with two branch points should always point from a thicker edge to a thinner one.

3.2.8 Segments atlas registration

An automatic way of assigning segment labels to portal vessels is to register an already labeled volume (atlas) to the actual volume, and copy the segment information from the atlas to the label map of the actual image.

3.2.9 Labeling of liver volume

Having the portal vein structure labeled with 8 segment labels, the whole segmented liver volume should be labeled with the 8 segment labels.

Fast marching algorithm is started from seed points being the manually labeled vessel points. Our modified fast marching algorithm iterates through every voxel of the image volume with constant speed, and labels every voxel with the nearest segment label. This way, the liver voxels and closest vessel voxels will have the same label.

4 Results

The proposed framework was implemented using the free MITK software environment. The prototype allows running the proposed functionalities in a sequence. After each step the results can be visualized in 2D and 3D views and the user can make manual correction before starting the subsequent module.

This work is currently in research prototype state thus only limited and early evaluation is available. The most important criteria for portal vein segmentation that the result must contain at least the secondary branches to reduce the volumetric error of segment separation under 10%.

Tested on 20 (healthy 4, tumors 16) contrast-enhanced CTs with manual liver contour. Automatic portal vein segmentation fulfills the criteria in 30% of the cases, with manual portal vein seed correction the result is 55%. The figures below illustrate a case where the algorithm finds the seed point of the portal vein, segments at least third level branches and automatically labels the segments.



Figure 13. Segmented liver and the portal vein (automatically segmented and labeled)

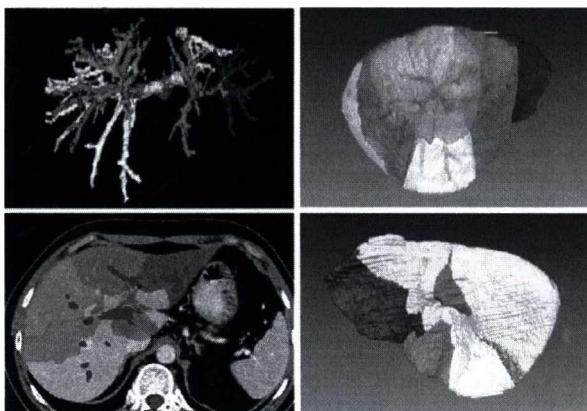


Figure 14. Individual liver segments

Missing of main portal vein branches is a significant issue, which can be caused by

- vasoconstriction or partial volume artifact, where the algorithm misclassifies the following vessel segment as a part of the hepatic vessel
- incorrect portal root, where the algorithm mistakenly identifies the portal root after the first branching point

The main direction should be the improvement of the accuracy of the portal vein segmentation. The figure below illustrates the most common problems found through the testing. In the left picture a main portal branch is missing (dark gray) from the segmentation (white). On the right picture a part of the hepatic vein is misclassified as portal vein.

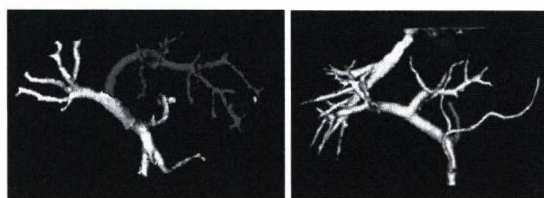


Figure 15. Common problems: missing portal branch (left) hepatic vessel included (right)

5 Conclusion and future directions

In this article we presented an automatic framework for portal vein segmentation and anatomical segment separation. It can segment the vessel structure of the liver, detect the root of the portal vein, separate the hepatic and portal structure, use an atlas for labeling branches and calculate the liver segments.

Preliminary results show the method is promising, but evidently, future work on the methods and extensive medical validation are needed. It is important to note, that the result of the different modules require manual correction in most of the cases, so future work shall also focus on the robustness of the algorithms and the elimination of user interactions.

6 References

- [1] Radtke A, Sgourakis G, Sotiropoulos GC, Molmenti EP, Nadalin S, Fouzas I, Schroeder T, Saner FH, Schenk A, Cicinnati VR, Malagó M, Lang H. Hepatic hilar and sectorial vascular and biliary anatomy in right graft adult live liver donor transplantation. *Transplant Proc.* 2008 Nov;40(9):3147-50.
- [2] Selle D, Preim B, Schenk A, Peitgen HO. Analysis of vasculature for liver surgical planning. *IEEE Trans Med Imaging.* 2002 Nov;21(11):1344-57.
- [3] Liver: Segmental Anatomy. Robin Smithuis
- [4] Atasoy C, Ozyürek E. Prevalence and types of main and right portal vein branching variations on MDCT. *AJR Am J Roentgenol.* 2006 Sep;187(3):676-81.
- [5] Koç Z, Oğuzkurt L, Uluşan S. Portal vein variations: clinical implications and frequencies in routine abdominal multidetector CT. *Diagn Interv Radiol.* 2007 Jun;13(2):75-80.
- [6] Arora J, Kapur V, Kakkar A, Dixit PC. Ramification Pattern Of Portal Vein In Right Lobe of Liver - A Corrosion Cast Study. *J Anat. Soc. India.* 2003 52(1) 12-14.
- [7] R. Beichel, T. Pock, Ch. Janko, R. Zotter, B. Reitingner, A. Bornik, K. Pal'agyi, E. Sorantin, G. Werkgartner, H. Bischof, and M. Sonka. Liver segment approximation in CT data for surgical resection planning. In *Medical Imaging 2004, Proceedings of SPIE*, pages 1435–1446, San Diego, 2004.
- [8] Soler L, Delingette H, Malandain G, Montagnat J, Ayache N, Koehl C, Dourthe O, Malassagne B, Smith M, Mutter D, Marescaux J. Fully automatic anatomical, pathological, and functional segmentation from CT scans for hepatic surgery. *Comput Aided Surg.* 2001;6(3):131-42.
- [9] Kawajiri S, Zhou X, Zhang X, Hara T, Fujita H, Yokoyama R, Kondo H, Kanematsu M, Hoshi H. Automated segmentation of hepatic vessels in non-contrast X-ray CT images. *Radiol Phys Technol.* 2008;1:214–222.
- [10] Fridman Y, Pizer S, Aylward S, Bullitt E. Extracting branching tubular object geometry via cores *Medical Image Analysis*, 8(3), 2004 pp. 169-176.
- [11] Palagyi, K., Tschirren, J., Hoffman, E.A., Sonka, M.: Quantitative analysis of pulmonary airway tree structures, *Computers in Biology and Medicine* 36, 2006, 974-996.
- [12] Sato Y, Nakajima S, Atsumi H, Koller T, Gerig G, Yoshida S, Kikinis R. 3D Multi-scale line filter for segmentation and visualization of curvilinear structures in medical images. *Lecture Notes In Computer Science*, 1997;1205:213-222.
- [13] Lee T, Kashyap RL, Chu C. Building skeleton models via 3-D medial surface/axis thinning algorithms. *CVGIP: Graph. Models Image Process.* 1994 Nov;56(6): 462-478

High-Speed Implementation of Bleeding Detection in Capsule Endoscopy Images using GPGPU Computing

Ágoston Srp and Ferenc Vajda

Department of Control Engineering and Information Technology
Budapest University of Technology and Economics, Budapest, Hungary

Abstract

Medical image processing always had the problem of implementing experimental algorithms in a manner suitable for day-to-day use. Required computation time always was and is a major constraint preventing the application of many promising algorithms. The emergence of today's GPUs capable of general purpose computing heralds a new age in the application of computing-intensive algorithms in daily use. This paper demonstrates the capability of these GPUs in decreasing the computing time of current medical image processing algorithms to acceptable levels. A bleeding detection algorithm for wireless capsule endoscopy images is used for demonstration purposes, as this is one of the fields where automation is inescapable due to the sheer number of images to be examined. The implementation of the algorithm in DirectX 9 and 10 brought expected, but still astonishing results: computation time decreased by nearly an order of magnitude.

Categories and Subject Descriptors (according to ACM CCS): I.4.6 [Image Processing and Computer Vision]: Pixel Classification

1. Introduction

Graphics processors (GPUs) are emerging as powerful computational co-processors for general purpose computations. The demands of graphics as well as non-graphics applications have driven GPUs to be today's most powerful computational hardware for the dollar¹.

Since GPUs are specialized for computation-intensive highly-parallel applications (e.g. graphics rendering), unlike CPUs, GPUs devote more transistors to data processing rather than data caching and flow control. Current GPUs are capable of about ten times as many GFLOPS[†] as current CPUs. GPU computational power doubles every ten months (surpassing the Moore's Law for traditional microprocessors) whereas CPU computational power doubles every seventeen months.

One of the main problems in medical image processing is the time constraint. It is simply not suitable to employ algorithms requiring an excessive amount of computing time, their superiority notwithstanding. Other fields pose a

challenge with the sheer number of pictures to be examined, quickly rocketing computing times to unacceptable levels. This is especially apparent in the case of wireless capsule endoscopy (WCE). A typical WCE examination result consists of between 45-50 **thousand** images, each having to be examined. The enormous number of images requires automation, as physician time is very expensive and with the number of images to be examined fatigue is a significant issue, compromising diagnosis reliability.

A supreme example for the prevailing problems is automated bleeding detection in WCE images. On one hand all possible bleedings have to be identified, as the purpose is to reduce the workload of the diagnosing physician and "discarded" images (i.e. images marked as non-bleeding) will not be examined further. Falsely discarded images containing bleeding can lead to disaster, as they may even result in patient fatality. As permitting such a scenario is unacceptable, every effort must be taken to prevent it. Ideally this requires 100% reliability regarding negative results and results in relatively minor

[†] Giga Floating Point Operations Per Second

importance of the lowest possible number of false positive identifications. Low number of these is still a priority but far less important than absolute certainty in non-bleeding classification. False positives, while to be avoided, are not harmful in small numbers. However, care must be taken, as a large number of false positives defeats the whole purpose of the algorithm. In practice the absolute requirements for non-bleeding classification may be relaxed a little without compromising effectiveness and reliability. As the same bleeding is usually visible in multiple WCE images due to the relatively high frequency of imaging (multiple pictures per second, the exact number dependent on the type of capsule) relative to the movement speed of the capsule in the bowel, it is sufficient to identify a single picture of such a group as positive bleeding. The diagnosing physician will of course examine the images adjacent to the one identified as bleeding to explore the bleeding site and its surroundings. The overall goal is to indicate all bleeding sites, with the smallest possible number of false bleeding classifications. The first is necessary for safe and responsible application, while the second reduces the workload of physicians further and is also desirable to inspire "trust" in the procedure so that even very small bleedings and/or lesions can be identified and measures can be taken before they become a "big" issue.

2. The Implemented Algorithm

Multiple approaches have been tried in bleeding detection. Just to name a few, Kodogiannis and Lygouras³ proposed an integrated decision-support system. Their approach identifies texture features in the texture spectrum in chromatic and achromatic regions on a selected area from the histogram of the colour components with a complex neural network. Bourbakis⁴ advocated the use of synergistically integrated image processing, - analysing and identifying procedures to detect anomalies. Coimbra and Cunha investigated the feasibility of MPEG-7 visual descriptors regarding automated image interpretation.

The example used in this paper is the algorithm proposed by Lau and Correia², which is very attractive both in detection rate and the number of false positives (if the parameters are suitably chosen).

The only negative aspect is the required computing time. (Their algorithm works in HSV space, requiring a conversion from RGB to HSV representation for every image regrettably increasing computing time considerably.) Further of note is that the algorithm uses per pixel criteria to detect the presence of bleeding, so GPU computing promises a considerable decrease in computing time.



Figure 1: Source image

The original algorithm utilises a preliminary qualifying function to reduce the number of images that have to be examined on a per-pixel basis. This function divides the image into 16 blocks and examines the average pixel intensity in each. This function while capable of speeding up computation on CPUs would necessitate either doing this computation on CPU, or for every image to be run twice through the graphics pipeline. This is not acceptable and on average only increases required computation time if GPU processing is implemented, thus we decided to omit this function. The main qualifying function examines the Saturation and/or Intensity values of each pixel in relation to the average intensity of the image to decide if it corresponds to bleeding. The image itself is qualified as bleeding if the number of bleeding pixels surpasses a threshold value.

To summarize: Two per-pixel operations have to be performed on each image; conversion from RGB to HSV representation, and bleeding detection. We further need to compute the average pixel intensity for each image. The per-pixel operations can be massively parallelised in the graphic pipeline.

An important fact is that the PillCam system forming the basis of our research stores the images in an avi container as jpeg images. Using traditional means, these had to be decompressed using the Independent JPEG Group's libjpeg library requiring further processing time. Contrary, using GPU processing an image can be loaded into a texture while generating its mipmap⁶. The mipmap is generated using box filtering. Box filtering is used to calculate the new pixel values when decreasing resolution by a factor of two. The resulting pixel is values are calculated by averaging the original pixel and its "east", "south" and "southeast" neighbours (Figure 2). This causes the lowest mipmap level to be the exact average of all pixels of the image. The algorithm requires the computation of the average pixel intensity for every image so we can take massive advantage of the built in functionality:

Automatically generating mipmaps at texture creation time takes advantage of hardware filtering because the mipmap resides in video memory. This further speeds up the algorithm.

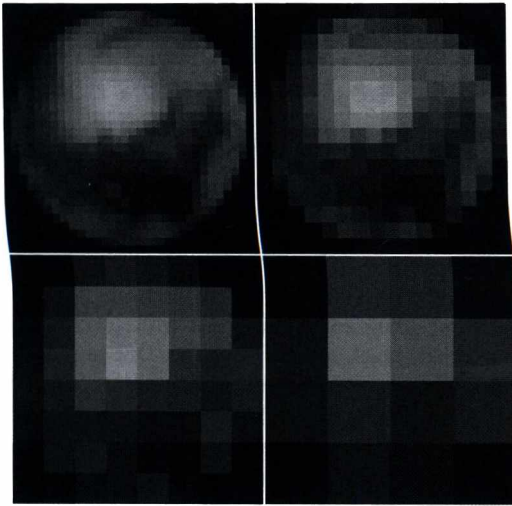


Figure 2: Mipmap levels 3, 4, 5 and 6 (the original image is level 0)

At this point we have the picture (Figure 1) and its full mipmap[†] (Figure 2) in video memory as a texture, its lowest level representing the average values for the image. An object to apply the texture to is also needed, a simple rectangle is sufficient for our purposes.

It is imperative to render to a texture and to avoid rendering anything onto the main display, as this would limit execution speed considerably. After ensuring this, the whole bleeding detection can be done in a single pass using vertex and pixel shaders 3.0 (DirectX 9 implementation) or 4.0 (DirectX 10 implementation) utilising the effect interface. Using lower version shaders is not possible due to the limited number of instructions allowed in early shader models.

The vertex shader is used to access the lowest mipmap level of the texture, extracting the average of the RGB values of the image and to facilitate the conversion from RGB to HSV to gain the average pixel intensity, which in turn is then passed on to the pixel shader. The algorithm however does not utilise the H value (Hue). A further speed-up was gained by simply omitting the computation of the Hue. As a rectangle has 4 vertices the conversion of the average is done 4 times. It is supremely important to do

[†] A mipmap is considered full, or complete, if the dimension of the lowest image of the collection is 1x1 pixel and all previous maps are present

the RGB-HSV conversion of the average in the vertex shader, as otherwise this calculation would be done for every pixel (256x256 times). This waste of processing time can not be tolerated.

The data is then passed onto the pixel shader in the graphics pipeline. The average value is passed on as an additional COLOR attribute, represented in HSV space. The RGB-HSV conversion is applied to every pixel before the bleeding detection function is run. Aforementioned function simply realises the main qualifying function as proposed by Lau and Correia. The pixel shader then passes on the color "white" for the pixel if it is found to be bleeding, and "black" if not (Figure 3).



Figure 3: Bleedings map

The result of this rendering step is a texture containing a map of bleeding pixels. The number of these is needed to decide if the image is to be classified as bleeding. There are two possibilities at this point:

Counting the white pixels in the image using the CPU may be relatively easy to realise, however it is not very efficient to put it mildly.

The other option is to apply the same technique used earlier to this texture. There are 3 possibilities: First is generating a mipmap of the result using automatic mipmap generation. The second achieves the same end result using the `IDirect3DDevice9::StretchRect` command successively with `D3DTEXF_LINEAR` as filter-option to ensure suitable filtering, decreasing resolution by half until 1x1 pixel resolution is reached. This function however is NOT available in DirectX 10, instead a separate shader doing this exact computation has to be realised. These 3 are equivalent solutions, as automatic mipmap generation simply packages the process into a single command for convenience's sake.

Whichever method used, the end result is a 1x1 pixel surface which contains the average value of the result, which in turn is proportional to the number of bleeding

pixels. These can be obtained if required by multiplying the aforementioned with the number of pixels in an image. It is supremely important to note at this point, that extreme care has to be taken when accessing the results. Accessing these surfaces in DirectX 10 requires mapping them to be accessible to the CPU. Accessing this mapped result in an unsuitable manner will massively slow down the processing. If access is premature the command will wait for the result to become available after the rendering is finished, thus causing a pipeline stall in the GPU, as no new commands will be fetched to the command buffer in the meantime. To prevent this it is recommended to copy the result to a staging resource and wait at least 2 frames before attempting access. Fortunately this is simply achieved by implementing a small circular buffer.

3. Results

The relevant specifications of the computer used for the benchmarking of the algorithm are as follows: AMD Turion64 X2 Mobile TL-50 1.6 GHz CPU, and 1024MB RAM.

The modified algorithm adapted for GPU processing was tested on a different PC. Specifications of the new system: Intel Core Duo E7300 2.66 GHz CPU, 2048 RAM and NVIDIA GeForce 9800GT graphics card.

As the CPU involvement or RAM usage in GPU processing is negligible the comparison is still viable.

Our test verified a considerable decrease in computing time (Table 1), to the point where everyday application is feasible. The superiority of the DirectX 10 implementation is evident, however even the DirectX 9 implementation still offers a considerable decrease in processing time. DirectX 10 implementation offers a processing speed of around 204 FPS[†].

4. Conclusions

This paper demonstrates the capability of GPU processing to massively reduce computation time of medical image processing algorithms to a fraction of their former value. The example of bleeding detection in wireless capsule endoscopy images was used, as this is a particularly computation intensive task due to the sheer number of images involved. Different bleeding detection algorithms were introduced and the one chosen for the demonstration was presented in some detail. During the presentation several issues and points of particular influence on the end result of the GPU implementation contrary to a traditional CPU implementation were pointed out .

[†] Frames Per Second

Exam #	1	2	3	4
Images	57547	56761	53326	59764
CPU time	2697 s	2601 s	2485 s	2742 s
DX9 GPU time	357 s	335 s	325 s	377 s
DX9 CPU/GPU	7.5546	7.7642	7.6462	7.2732
DX10 GPU time	265 s	255 s	260 s	293 s
DX10 CPU/GPU	10.17	10.20	9.55	9.3584

Table 1: Test results

GPU computing offers a viable possibility to introduce algorithms into everyday medical practice heretofore unsuitable regarding computing time, while also opening up a whole new level in patient comfort. Due to increased computation capacity and reduced computation time it is possible for the diagnosis to be finished in minutes. The patient will merely be asked to wait a few minutes outside for the diagnosis to be finished. This elevates patient comfort hugely on multiple levels. No longer need the patient be sent back home to anxiously wait for the result of the examination, reducing stress. Also no additional travel arrangements have to be made for patients from remote regions just to receive the results.

References

1. J. D. Owens, D. Luebke, N. Govindaraju, M. Harris, J. Krüger, A. E. Lefohn, and T. J. Purcell: "A survey of general-purpose computation on graphics hardware" *Computer Graphics Forum*, 26(1):80–113, 2007.
2. P. Y. Lau, and P. L. Correia: „Detection of bleeding patterns in WCE video using multiple features" *Proceedings of the 29th Annual International Conference of the IEEE EMBS*; Cité Internationale, Lyon, France; August 23-26, 2007, Page(s): 5601 - 5604
3. V.S. Kodogiannis and J.N. Lygouras: „A Computerised Diagnostic Decision Support System in Wireless-Capsule Endoscopy", *Intelligent Systems, 2006 3rd International IEEE Conference on*, Sept. 2006, Page(s): 638 - 644
4. N. Bourbakis: „Detecting abnormal patterns in WCE images", *Bioinformatics and Bioengineering, 2005. BIBE 2005. Fifth IEEE Symposium on*, 19-21 Oct. 2005, Page(s): 232 - 238
5. M.T. Coimbra, J.P.S. Cunha: „MPEG-7 Visual Descriptors—Contributions for Automated Feature Extraction in Capsule Endoscopy", *Circuits and Systems for Video Technology, IEEE Transactions on* , May 2006, Volume: 16 , Issue: 5, Page(s): 628–637
6. L. Szirmay-Kalos, L. Szécsi, M. Sbert: "GPU-Based Techniques for Global Illumination Effects", Morgan and Claypool Publishers, San Rafael, USA, 2008

Iterative 3D Reconstruction with Scatter Compensation for PET-CT on the GPU

Milán Magdics, Balázs Tóth, and Ádám Csentesi

BME IIT

Abstract

This paper presents a fully 3D reconstruction algorithm for Positron Emission Tomography (PET) running on the Graphics Processing Unit (GPU). Using the electron density reconstructed by an additional Computer Tomograph, single and multiple scattering effects are also estimated and compensated. We propose gathering type forward- and back-projection steps executed by the GPU with no GPU-CPU communication. The error analysis leads us to the conclusion that forward-projection should be computed with similar number of samples on all detector pairs. With these advancements, over 256^3 resolution voxel arrays can be reconstructed in a few minutes.

1. Introduction

In positron emission tomography (PET) we need to find the spatial intensity distribution of positron-electron annihilations. During an annihilation event, two oppositely directed 511 keV photons are produced². During measurement we collect the number of simultaneous photon incidents in detector pairs, also called *Lines Of Responses* or *LORs*. Before being detected in the detectors, photons might scatter, get absorbed due to the *photoelectric effect*, or produce new photon pairs, but in our energy range and for living organs only scattering is relevant. The probability of scattering in unit distance is the *scattering cross section* σ . When scattering happens, there is a unique correspondence between the relative scattered energy and the cosine of the scattering angle, as defined by the *Compton formula*:

$$\epsilon = \frac{1}{1 + \epsilon_0(1 - \cos \theta)},$$

where $\epsilon = E_1/E_0$ expresses the ratio of the scattered energy E_1 and the incident energy E_0 , and $\epsilon_0 = E_0/(m_e c^2)$ is the incident photon energy relative to the energy of the electron.

The differential of the *scattering cross section*, i.e. the probability density that the photon is scattered from direction $\vec{\omega}'$ into differential solid angle $d\omega$ in direction $\vec{\omega}$, is given by the *Klein-Nishina formula*²⁰:

$$\frac{d\sigma(\vec{v}, \cos \theta, \epsilon_0)}{d\omega} = \frac{r_e^2 C(\vec{v})}{2} (\epsilon + \epsilon^3 - \epsilon^2 \sin^2 \theta),$$

where $\cos \theta = \vec{\omega} \cdot \vec{\omega}'$, $C(\vec{v})$ is the electron density, and $r_e =$

$2.82 \cdot 10^{-15}$ [m] is the classical electron radius. The Klein-Nishina formula defines the product of the scattering cross section and the conditional probability density of the scattering direction. The scattering cross section can be obtained as the directional integral of the Klein-Nishina formula over the whole directional sphere:

$$\sigma(\vec{v}, \epsilon_0) = \int_{\Omega} \frac{d\sigma(\vec{v}, \cos \theta, \epsilon_0)}{d\omega} d\omega = \frac{r_e^2 C(\vec{v})}{2} \sigma^0(\epsilon_0) \quad (1)$$

where Ω is the directional sphere and σ^0 is the *normalized scattering cross section*:

$$\sigma^0(\epsilon_0) = \int_{\Omega} \epsilon + \epsilon^3 - \epsilon^2 \sin^2 \theta d\omega = 2\pi \int_{-1}^1 \epsilon + \epsilon^3 - \epsilon^2 \sin^2 \theta d \cos \theta.$$

The ratio of the Klein-Nishina formula and the scattering cross section is the *phase function*, which defines the probability density of the reflection direction, provided that reflection happens:

$$P_{KN}(\cos \theta, \epsilon_0) = \frac{d\sigma}{d\omega} / \sigma = \frac{\epsilon + \epsilon^3 - \epsilon^2 \sin^2 \theta}{\sigma^0(\epsilon_0)}.$$

In a PET combined with a Computer Tomograph (CT), two measurements on different energy levels are made. First, the CT calculates the total extinction of the gamma photons along the LORs, from which electron density $C(\vec{v})$ is reconstructed. Then, the PET reconstruction can take advantage of this information to improve its accuracy.

2. PET reconstruction problem

The inputs of the reconstruction algorithm are the measured responses of detector pairs, called LORs: $(y_1, y_2, \dots, y_{N_{LOR}})$. Additionally, we also know the electron density $C(\vec{v})$ in the volume, which is measured by the CT equipment.

The required output of the reconstruction method is the *emission density* function $x(\vec{v})$ that describes the number of photon pairs (i.e. the annihilation events) born in a unit volume around point \vec{v} . To represent the unknown function with finite data, it is approximated in a *finite element* form:

$$x(\vec{v}) = \sum_{V=1}^{N_{\text{voxel}}} x_V b_V(\vec{v}),$$

where $x_1, x_2, \dots, x_{N_{\text{voxel}}}$ are the unknown coefficients and $b_V(\vec{v})$ ($V = 1, \dots, N_{\text{voxel}}$) are pre-defined basis functions. For example, if $b_V(\vec{v})$ is constant and equal to the reciprocal of the voxel volume in voxel V and zero otherwise, then the finite element approximation is piece-wise constant. On the other hand, if $b_V(\vec{v})$ tri-linearly decreasing from the voxel center to the centers of the neighboring voxel, then we work with tri-linear approximation. We assumed that the basis functions are normalized, that is $\int_{\mathcal{V}} b_V(\vec{v}) d\mathbf{v} = 1$ where \mathcal{V} is the volume of interest. Note that normalization means that basis function $b_V(\vec{v})$ is a probability density. In this paper we use *tri-linear* basis functions because it is directly supported by the graphics hardware so it has no additional computational cost.

The correspondence between the coefficients of the emission function (voxel intensities) and the detector responses is built of the elemental conditional probability density that a photon pair lands at detector points \vec{z}_1 and \vec{z}_2 given that they are emitted at point \vec{v} in directions $\vec{\omega}$ and $-\vec{\omega}$, which is denoted by

$$P(\vec{v}, \vec{\omega} \rightarrow \vec{z}_1, \vec{z}_2).$$

This probability depends on the level of accuracy on which the physical phenomena are simulated:

- *Without scatter compensation*, this density is zero if either \vec{z}_1 or \vec{z}_2 is not on the line going through \vec{v} and of direction $\vec{\omega}$. If they are on the line, this density equals to the probability that no extinction happens along the line, which also depends on photon energy ϵ_0 :

$$P(\vec{v}, \vec{\omega} \rightarrow \vec{z}_1, \vec{z}_2) = e^{-\int_{\vec{z}_1}^{\vec{z}_2} \sigma(\vec{l}, \epsilon_0) dl}.$$

In our particular case, the attenuation and detector probability should be evaluated at $\epsilon_0 = 1$, i.e. at the energy level of the electron, since now we consider only photons that have not scattered yet.

- *With scatter compensation*, the forward-projection also simulates multiple scattering effects, making this probability equal to the probability that the pair of photons

arrived at \vec{z}_1 and \vec{z}_2 after zero, one, two, etc. scattering events. As scattering may happen anywhere, the contribution of single scattering is a 3D integral, the contribution of double scattering is a 6D integral, etc. In theory, the probability involving arbitrary number of scattering points can be expressed as an infinite dimensional integral.

Assuming that detectors are ideal — i.e. that a photon arriving at the surface of a detector may contribute only this detector and may not get scattered into another detector crystal — the probability that LOR L detects a photon pair is the integral of the arrival density for point pairs on these detectors of areas D_1 and D_2 , multiplied with the probability that the photons are detected:

$$P(\vec{v}, \vec{\omega} \rightarrow L) =$$

$$\int_{D_1} \int_{D_2} P(\vec{v}, \vec{\omega} \rightarrow \vec{z}_1, \vec{z}_2) \mathcal{D}_{\epsilon_0^{(1)}}(\vec{z}_1, \vec{\omega}) \mathcal{D}_{\epsilon_0^{(2)}}(\vec{z}_2, \vec{\omega}) dz_1 dz_2$$

where \mathcal{D}_{ϵ_0} is the energy dependent probability that the detector detects the photon, called *detector sensitivity*. In the general case, this probability also depends on the photon energy, the location and the direction of the incidence.

If a photon pair is isotropically emitted from point \vec{v} , then the probability that it is detected by LOR L is

$$P(\vec{v} \rightarrow L) = \int_{\Omega_H} P(\vec{v}, \vec{\omega} \rightarrow L) \frac{d\omega}{2\pi}.$$

We have to integrate only on the half sphere (Ω_H) having solid angle 2π since the two photons of the pair may be exchanged.

The total contribution from all points \vec{v} to the expected number of hits in LOR L is:

$$\tilde{y}_L = \int_{\mathcal{V}} x(\vec{v}) P(\vec{v} \rightarrow L) d\mathbf{v} = \sum_{V=1}^{N_{\text{voxel}}} x_V \int_{\mathcal{V}} b_V(\vec{v}) P(\vec{v} \rightarrow L) d\mathbf{v}.$$

This equation can also be written in a matrix form:

$$\tilde{\mathbf{y}} = \mathbf{A} \cdot \mathbf{x}$$

where the system response is characterized by a *system matrix* \mathbf{A} ³, which defines the correspondence between voxel intensities $\mathbf{x} = (x_1, x_2, \dots, x_{N_{\text{voxel}}})$ and expected LOR values $\tilde{\mathbf{y}} = (\tilde{y}_1, \tilde{y}_2, \dots, \tilde{y}_{N_{LOR}})$. The meaning of matrix element

$$\mathbf{A}_{LV} = \int_{\mathcal{V}} \int_{2\pi} b_V(\vec{v}) P(\vec{v}, \vec{\omega} \rightarrow L) \frac{d\omega}{2\pi} d\mathbf{v} \quad (2)$$

is the probability that a photon pair born in a random point distributed with density $b_V(\vec{v})$ is detected by LOR L .

The task of the reconstruction is to find voxel intensities of \mathbf{x} based on the measured LOR incidents in $\tilde{\mathbf{y}}$. Such problems can be attacked by iterative approaches that iterate *forward-*

projection

$$\tilde{y}_L = \sum_{V=1}^{N_{\text{voxel}}} \mathbf{A}_{LV} x_V^{(n)}, \quad (3)$$

and then correct $\mathbf{x}^{(n+1)}$ from $\mathbf{x}^{(n)}$ according to the similarity of the resulting guess of the expected value $\tilde{\mathbf{y}}$ and the measured response \mathbf{y} .

For example, based on *expectation maximization* (EM)¹¹, we obtain the following correction, also called *back-projection* scheme:

$$\frac{x_V^{(n+1)}}{x_V^{(n)}} = \frac{1}{\sum_{L=1}^{N_{\text{LOR}}} \mathbf{A}_{LV}} \cdot \sum_{L=1}^{N_{\text{LOR}}} \mathbf{A}_{LV} \frac{y_L}{\tilde{y}_L}. \quad (4)$$

In the forward- and back-projection steps, matrix elements \mathbf{A}_{LV} need to be estimated on-the-fly since neither their pre-computation nor their storage is feasible in our case, because of the following reasons:

- If scattering were ignored, the system matrix would be sparse. However, with scatter simulation all matrix elements can be non-zero, thus sparse matrix techniques and matrix compression do not work.
- The scattering parameters depend on the electron density function of the measured object, which is obtained by the CT in parallel to the PET measurement. Thus, the system matrix is different for every different object, so a single off-line simulation or measurement is not suitable for its computation.

The physically based 3D PET reconstruction problem is challenging. Iteration should work with very large matrices. When scattering is also considered, this matrix is not sparse. Due to the dependence of the scattering parameters of the measured object, the matrix cannot be pre-computed or measured, neither can the inherent geometric symmetries of the detector grids be exploited. A matrix element describes the effect of all possible paths of a photon pair originating in a voxel and landing at a detector pair, which may have arbitrary number of scattering points. Mathematically, the matrix element is an infinite dimensional integral, and in forward-projection the matrix elements must have sufficient precision. Iterating with such a huge matrix and re-computing matrix elements as high-dimensional integrals in each iteration step need enormous computation power if we wish to obtain the reconstruction results in reasonable time (i.e. at most in a few minutes).

3. Previous work

The time-consuming process of fully 3D iterative tomography reconstruction has been targeted by both algorithmic improvements and by looking for more efficient computational platforms. Among the high-performance computing possibilities, like FPGAs⁶, multi-CPU systems¹⁰, the

CELL processor⁴, and GPUs¹⁹, the GPU has proven to be the most cost-effective platform for such tasks⁷. GPUs can be programmed with two different programming models. Shader APIs like OpenGL/Cg or Direct3D/HLSL present the GPU hardware as the direct implementation of the incremental rendering pipeline¹⁴, including both programmable and fixed processing stages. On the other hand, CUDA⁸, StreamSDK, and OpenCL provide an access to the multiprocessors of the GPU where each multiprocessor contains a set of scalar processors organized into *warps* sharing the instruction unit, and therefore acting as a SIMD hardware.

Most of the GPU approaches published so far considered only the geometry in projection calculation and used the shader API for implementation^{1,19,9}. The rasterizer and the alpha-blending units accessible through the shader API support these simple calculations, but when the algorithm gets more complex, the incremental rendering pipeline view of the shader API becomes too restrictive and less intuitive. While attenuation correction and the incorporation of the point spread function in SPECT are relatively straightforward¹⁶, physically plausible scatter correction should be replaced by a simple blurring operation to stay within the constraints of the incremental rendering pipeline¹⁸.

However, when more complex algorithms are implemented, the additional control of the shader processors out-weights the possibility of exploiting the fixed function pipeline elements, like clipping, rasterization, depth-buffering or alpha blending. Considering these, we have chosen CUDA as the implementation platform.

Physically plausible scatter correction needs photon transport simulation and the evaluation of high-dimensional integrals in photon path space. As classical quadrature rules fail in higher dimensions due to the *curse of dimensionality*, these high-dimensional integrals are estimated by Monte Carlo or quasi-Monte Carlo methods¹³. Better results can be obtained for the single scattering case when the integral over the path space can be transformed to a volumetric integral¹⁷. Our method also uses this transformation, but extends it for multiple scattering. Furthermore, instead of random sampling, we apply deterministic Sigma-Delta modulation for sample generation¹⁵.

4. Reconstruction algorithm

4.1. Forward-projection without scatter correction

Forward projection computes the detector responses from the current emission density estimation defined by vector $\mathbf{x} = (x_1, x_2, \dots, x_{N_{\text{voxel}}})$:

$$\tilde{y}_L = \sum_{V=1}^{N_{\text{voxel}}} \mathbf{A}_{LV} x_V = \int \int_{\mathcal{V}} x(\vec{v}) P(\vec{v}, \vec{\omega} \rightarrow L) \frac{d\omega}{2\pi} dv.$$

If we ignore photon scattering, then a LOR can be affected only if its detectors are seen at directions $\vec{\omega}$ and $-\vec{\omega}$

from emission point \vec{v} . It also means that emission point \vec{v} and direction $\vec{\omega}$ unambiguously identify detector hit points \vec{z}_1 and \vec{z}_2 , or alternatively, from detector hit points \vec{z}_1 and \vec{z}_2 , we can determine those emission points \vec{v} and direction $\vec{\omega}$, which can contribute. We shall modify our view point from the emission points and directions to detector points, and using the correspondence between them, the detector response is expressed as an integral over the detector surfaces.

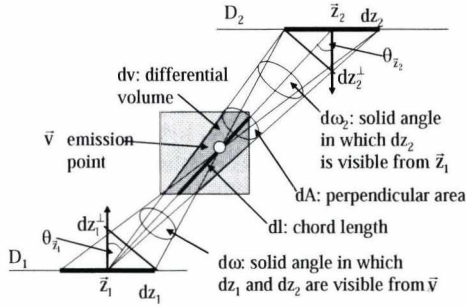


Figure 1: Computation of the Jacobian of the change of variables. The differential solid angle at which dz_1 detector surface and dz_2 detector surface are simultaneously seen from emission point \vec{v} is $d\omega = dz_1 \cos \theta_{z_1} / |\vec{z}_1 - \vec{v}|^2 = dz_2 \cos \theta_{z_2} / |\vec{z}_2 - \vec{v}|^2$. The differential solid angle at which dz_2 is seen from point \vec{z}_1 is $d\omega_2 = dz_2 \cos \theta_{z_2} / |\vec{z}_2 - \vec{z}_1|^2 = dA / |\vec{z}_1 - \vec{v}|^2$. Finally, the differential volume intersected by lines of \vec{z}_1 and \vec{z}_2 is $dv = dl dA$, where dl is the length of the line segment intersecting dv , and dA is the surface area that is perpendicular to the line.

According to Fig. 1, the Jacobian of the change of integration variables is:

$$\frac{d\omega}{2\pi} dv = \frac{d/dz_1^\perp dz_2^\perp}{2\pi |\vec{z}_1 - \vec{z}_2|^2} = \frac{\cos \theta_{z_1} \cos \theta_{z_2}}{2\pi |\vec{z}_1 - \vec{z}_2|^2} dz_1 dz_2.$$

where detector normals enclose angles θ_{z_1} and θ_{z_2} with the direction between \vec{z}_1 and \vec{z}_2 , thus the differential detector areas perpendicular to the LOR are $dz_1^\perp = \cos \theta_{z_1} dz_1$ and $dz_2^\perp = \cos \theta_{z_2} dz_2$. Let us introduce the *effective detector sensitivity* (the *detector's cross section*) as the product of detector sensitivity and the cosine of the angle between the detector normal and the incident direction:

$$\mathcal{D}_{\epsilon_0}^\perp(\vec{z}) = \mathcal{D}_{\epsilon_0}(\vec{z}) \cos \theta_{\vec{z}}.$$

If scattering is ignored, energy level $\epsilon_0 = 1$ is constant.

With this, the LOR integral can be expressed as:

$$\bar{y}_L = \int_{D_1} \int_{D_2} X(\vec{z}_1, \vec{z}_2) T_1(\vec{z}_1, \vec{z}_2) \mathcal{D}_1^\perp(\vec{z}_1) \mathcal{D}_2^\perp(\vec{z}_2) dz_1 dz_2, \quad (5)$$

where D_1 and D_2 are the surfaces of the two detectors of the

given LOR,

$$X(\vec{z}_1, \vec{z}_2) = \frac{1}{2\pi} \int_{\vec{z}_1}^{\vec{z}_2} x(\vec{l}) dl$$

is the total emission of photons along the line segment of end points \vec{z}_1 and \vec{z}_2 , and

$$T_{\epsilon_0}(\vec{z}_1, \vec{z}_2) = \frac{1}{|\vec{z}_1 - \vec{z}_2|^2} \cdot e^{-\int_{\vec{z}_1}^{\vec{z}_2} \sigma(\vec{l}, \epsilon_0) dl}$$

is the attenuation factor.

Equation (5) can be estimated by taking N_{deline} uniformly distributed point pairs, $(\vec{z}_1^{(i)}, \vec{z}_2^{(i)})$ on the two detectors:

$$\bar{y}_L \approx \frac{D_1 D_2}{N_{deline}} \sum_{i=1}^{N_{deline}} X^{(i)} T^{(i)} \mathcal{D}_1^\perp(\vec{z}_1^{(i)}) \mathcal{D}_2^\perp(\vec{z}_2^{(i)})$$

where $X^{(i)}$ and $T^{(i)}$ are estimated by *ray-marching* on the line segment $(\vec{z}_1^{(i)}, \vec{z}_2^{(i)})$ visiting points \vec{l}_{ij} in step j , being at distance Δl_j (Fig. 2):

$$X^{(i)} \approx \frac{1}{2\pi} \sum_{j=1}^{N_{march}} x(\vec{l}_{ij}) \Delta l_j.$$

$$T^{(i)} \approx \frac{1}{|\vec{z}_1^{(i)} - \vec{z}_2^{(i)}|^2} \cdot e^{-\sum_{j=1}^{N_{march}} \sigma(\vec{l}_{ij}, \epsilon_0) \Delta l_j}.$$

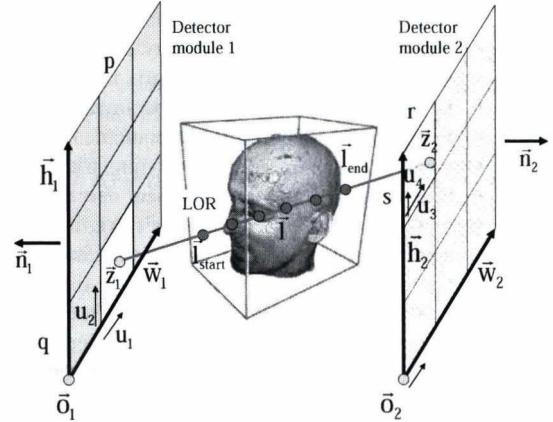


Figure 2: A single computational thread of forward-projection takes a detector pair and marches on the rays between the sample points of the detectors.

The error of this estimation depends on the number of line samples N_{deline} and point samples per line N_{march} , and also on the discrepancy of the samples in the five-dimensional space. As random samples are more uniform in

higher dimensional spaces than grids, we obtain 4D samples $(u_1^{(i)}, u_2^{(i)}, u_3^{(i)}, u_4^{(i)})$ randomly.

4.2. Back-projection

Back-projection evaluates equation (4), i.e. it computes a fraction for each voxel V , where the denominator and the numerator are

$$\text{Denom} = \sum_{L=1}^{N_{LOR}} \mathbf{A}_{LV}, \quad \text{Enum} = \sum_{L=1}^{N_{LOR}} \mathbf{A}_{LV} \frac{y_L}{\bar{y}_L}.$$

Back-projection can be executed approximately without changing the fixed point of the iterations scheme, thus, the volumetric integral of the system matrix is estimated from a single position sample \vec{v} obtained with a probability density of $b_V(\vec{v})$, and the directional integral is approximated by a single sample per detector d_1 , which subtends solid angle $\Delta\omega_{d_1}$ (Fig. 3). Emission point \vec{v} and point \vec{z}_1 on detector d_1 determines direction $\vec{\omega}$. If scattering is ignored, $P(\vec{v}, \vec{\omega} \rightarrow L)$ is non zero for that $L(d_1, d_2)$ LOR where the line of \vec{z}_1 and \vec{v} intersects detector d_2 . For this LOR, we get:

$$\mathbf{A}_{LV} = \int_{\mathcal{V}} \int_{\Omega_H} b_V(\vec{v}) P(\vec{v}, \vec{\omega} \rightarrow L) \mathcal{D}_1^2 \frac{d\omega}{2\pi} d\mathcal{V} \approx$$

$$\int_{\Omega_H} P(\vec{v}, \vec{\omega} \rightarrow L) \mathcal{D}_1^2 \frac{d\omega}{2\pi} \approx \frac{\Delta\omega_{d_1}}{2\pi} \mathcal{D}_1^2,$$

where \mathcal{D}_1 is the detector efficiency if the photon is on the energy level of the electron and it arrives perpendicularly at the middle of the detector. For other LORs associated with detector d_1 , $\mathbf{A}_{L(d_1, d_2), V} = 0$.

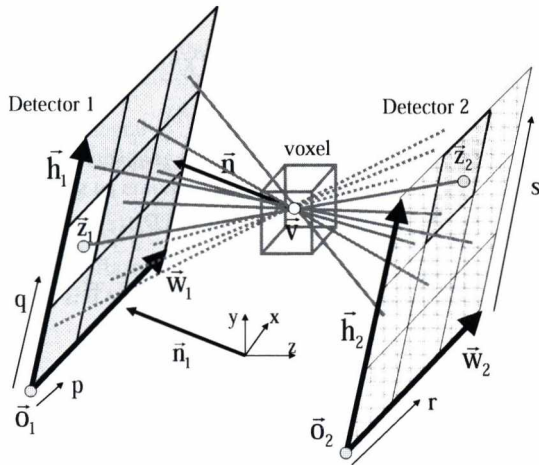


Figure 3: A single computational thread of back-projection takes an emission point sample \vec{v} and processes all LORs that can cross this sample point.

The directly sampled detector d_1 is called *primary*. The detector that is obtained by line intersection is the *secondary*. As secondary detector d_2 is unambiguously determined by voxel sample \vec{v} and primary detector sample \vec{z}_1 , it is a function of them, which is denoted by $d_2(\vec{z}_1, \vec{v})$. Note that not all primary detector – voxel pairs, i.e. (\vec{z}_1, \vec{v}) , correspond to a secondary detector. It can happen that the line defined by points \vec{z}_1 and \vec{v} does not intersect any modules, that is, it misses all detectors. When the integrals are estimated by discrete sums over the primary detectors, these detectors should be skipped:

$$\text{Denom} \approx \sum_{d_1=1, d_2 \text{ exists}}^{N_{det}} \frac{\Delta\omega_{d_1}}{2\pi} \mathcal{D}_1^2,$$

$$\text{Enum} \approx \sum_{d_1=1, d_2 \text{ exists}}^{N_{det}} \frac{\Delta\omega_{d_1}}{2\pi} \mathcal{D}_1^2 \frac{y_{L(d_1, d_2)(\vec{z}_1, \vec{v})}}{\bar{y}_{L(d_1, d_2)(\vec{z}_1, \vec{v})}}.$$

As we take the fraction of the two values, we can simplify with $\mathcal{D}_1^2/(2\pi)$.

Solid angle $\Delta\omega_{d_1}$ is the size of the directional set where primary detector d_1 is seen from voxel point \vec{v} . If the surface area of the detector is D_1 and the angle between the detector's unit normal vector \vec{n}_1 and the direction of $\vec{z}_1 \rightarrow \vec{v}$ is θ_{z_1} , then

$$\Delta\omega_{d_1} \approx \frac{D_1 \cos \theta_{z_1}}{|\vec{v} - \vec{z}_1|^2} = D_1 \frac{\vec{n}_1 \cdot (\vec{v} - \vec{z}_1)}{|\vec{v} - \vec{z}_1|^3}.$$

Assuming a fixed emission point \vec{v} , intersection point \vec{z}_2 is obtained from primary sample \vec{z}_1 as a *central projection*. Thus, \vec{z}_2 is a homogeneous linear function of \vec{z}_1 , and similarly (r, s) can be obtained as the integer part of a homogeneous linear functions of (p, q) :

$$r = \lfloor \frac{C_r + A_r p + B_r q}{\gamma - \alpha p - \beta q} \rfloor, \quad s = \lfloor \frac{C_s + A_s p + B_s q}{\gamma - \alpha p - \beta q} \rfloor.$$

The coefficients in this formula are determined from the detector geometry.

5. Accuracy improvement with scattering simulation

So far, we have ignored in-scattering, that is, we assumed that emission point \vec{v} and detector hit points \vec{z}_1 and \vec{z}_2 are on a line. With this assumption all scattered photons are lost. If we consider photon scattering, the path of the photon pair will be a *polyline* containing the emission point somewhere inside one of its line segments (Fig. 4). This polyline includes scattering points $\vec{s}_1, \dots, \vec{s}_S$ where one of the photons changed its direction in addition to detector hit points $\vec{z}_1 = \vec{s}_0$ and $\vec{z}_2 = \vec{s}_{S+1}$. The values measured by detector pairs will then be the total contribution, i.e. the integral of such polyline paths of arbitrary length.

We consider the contribution of photon paths as an integral over the Cartesian product set of the volume. This integration domain is sampled globally, i.e. a single sample

is used for the computation of all detector pairs. Sampling parts of photon paths globally and *reusing* a partial path for all detector pairs allow us to significantly reduce the number of samples and consequently the computation time.

To express the contribution of a polyline path, we take its line segments one-by-one and consider a line segment as a *virtual LOR* with two virtual detectors of locations, \vec{s}_{i-1} and \vec{s}_i , and of differential areas projected perpendicularly to the line segment, dA_{i-1}^\perp and dA_i^\perp (Fig. 4).

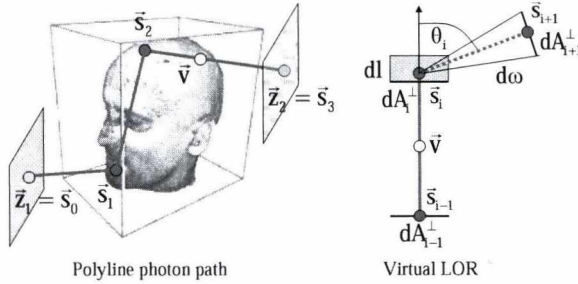


Figure 4: The scattered photon path is a polyline (left) made of virtual LORs (right). The left figure depicts the case of $S = 2$.

According to equation (5) the contribution of a virtual LOR at its endpoints, i.e. the expected number of photon pairs going through dA_{i-1}^\perp and dA_i^\perp is

$$\mathcal{X}_i \mathcal{T}_i dA_{i-1}^\perp dA_i^\perp,$$

where

$$\mathcal{X}_i = X(\vec{s}_{i-1}, \vec{s}_i), \quad \mathcal{T}_i = T_{\epsilon_0^{(i)}}(\vec{s}_{i-1}, \vec{s}_i).$$

In the line segment of the emission, the original photon energy has not changed yet, thus $\epsilon_0^{(i)} = 1$.

Suppose that scattering happens around end point \vec{s}_i of the virtual LOR in differential volume $ds_i = dA_i^\perp dl$, i.e. at run length dl (right of Fig. 4). Now let us extend this virtual LOR by a single scattering step forming a polyline $\vec{s}_{i-1}, \vec{s}_i, \vec{s}_{i+1}$. The probability of photon-electron annihilation along distance dl is $\sigma(\vec{s}_i, \epsilon_0^{(i)}) dl$. The photon scatters in solid angle $d\omega$ with probability $P_{KN}(\cos \theta_i, \epsilon_0^{(i)}) d\omega$ where θ_i is the scattering angle. The scattered photon will go along virtual LOR $(\vec{s}_i, \vec{s}_{i+1})$ with differential area dA_{i+1}^\perp at its end if area dA_{i+1}^\perp subtends solid angle $d\omega$, that is:

$$d\omega = \frac{dA_{i+1}^\perp}{|\vec{s}_i - \vec{s}_{i+1}|^2}.$$

Upon scattering the photon changes its energy to

$$\epsilon_0^{(i+1)} = \frac{\epsilon_0^{(i)}}{1 + \epsilon_0^{(i)} (1 - \cos \theta)}.$$

This photon arrives at the other end of this virtual LOR if

there is no further scattering inside this line segment, which is the case with probability $T_{\epsilon_0^{(i+1)}}(\vec{s}_i, \vec{s}_{i+1})$.

Summarizing, the expected number of photon pairs born between \vec{s}_{i-1} and \vec{s}_i and reaching differential areas dA_{i-1}^\perp and dA_{i+1}^\perp via scattering at differential volume ds_i is:

$$\mathcal{X}_i \mathcal{T}_i \mathcal{S}_i \mathcal{T}_{i+1} dA_{i-1}^\perp ds_i dA_{i+1}^\perp$$

where

$$\mathcal{S}_i = \sigma(\vec{s}_i, \epsilon_0^{(i)}) P_{KN}(\cos \theta_i, \epsilon_0^{(i)})$$

is the differential cross section of the scattering.

The integral of the contributions of paths of S scattering points is

$$\bar{y}_L^{(S)} = \int_{D_1} \int_{D_2} \int_{\mathcal{V}^S} \mathcal{D}_{\epsilon_0^{(1)}}^\perp \cdot \sum_{i=0}^S \mathcal{A}_i \cdot \mathcal{X}_{i+1} \mathcal{T}_{i+1} \cdot \mathcal{B}_{i+1} \cdot \mathcal{D}_{\epsilon_0^{(S)}}^\perp (d\mathbf{v})^S dz_2 dz_1, \quad (6)$$

where

$$\mathcal{A}_i = \prod_{j=1}^i \mathcal{T}_j \mathcal{S}_j$$

is the total attenuation between detector \vec{z}_1 and the i th scattering point, including the attenuation due to scattering at the end of the line segments, and

$$\mathcal{B}_{i+1} = \prod_{j=i+1}^S \mathcal{S}_j \mathcal{T}_{j+1}$$

is the total attenuation from between the $(i+1)$ th scattering point and detector point \vec{z}_2 , and $\epsilon_0^{(1)}$ and $\epsilon_0^{(S)}$ are the energies of the photons arriving at the first and second detectors, respectively.

For example, the integral of the contribution of paths of one scattering point is

$$\bar{y}_L^{(1)} = \int_{D_1} \int_{D_2} \int_{\mathcal{V}} dv dz_2 dz_1 \sigma(\vec{s}) P_{KN}(\cos \theta, 1) \cdot$$

$$[\mathcal{D}_1^\perp(\vec{z}_1) X(\vec{z}_1, \vec{s}) \mathcal{T}_1(\vec{z}_1, \vec{s}) T_{\epsilon_0}(\vec{s}, \vec{z}_2) \mathcal{D}_{\epsilon_0}^\perp(\vec{z}_2) +$$

$$\mathcal{D}_{\epsilon_0}^\perp(\vec{z}_1) T_{\epsilon_0}(\vec{z}_1, \vec{s}) X(\vec{z}_1, \vec{s}) \mathcal{T}_1(\vec{s}, \vec{z}_2) \mathcal{D}_1^\perp(\vec{z}_2)], \quad (7)$$

where ϵ_0 is the energy level of the photon after a single Compton scattering at angle θ . Scattering angle θ is formed by directions $\vec{s} - \vec{z}_1$ and $\vec{z}_2 - \vec{s}$.

When the attenuation is computed, we should take into account that the photon energy changes along the polyline and the scattering cross section also depends on this energy, thus different cross section values should be integrated when the annihilations on a different line segment are considered. As we wish to reuse the line segments and not to repeat ray-marching redundantly, each line segment is marched only once assuming photon energy $\epsilon_0 = 1$, and attenuation \mathcal{T}_1 for

this line segment is computed. Then, when the place of annihilation is taken into account and the real value of the photon energy ϵ_0 is obtained, initial attenuation T_1 is transformed. The transformation is based on the decomposition of equation (1):

$$\sigma(\vec{l}, \epsilon_0) = \sigma(\vec{l}, 1) \cdot \frac{\sigma^0(\epsilon_0)}{\sigma^0(1)}.$$

Using this relation, we can write

$$T_{\epsilon_0} |\vec{z}_1 - \vec{z}_2|^2 = e^{-\int_{\vec{z}_1}^{\vec{z}_2} \sigma(\vec{l}, \epsilon_0) d\vec{l}} = e^{-\frac{\sigma^0(\epsilon_0)}{\sigma^0(1)} \int_{\vec{z}_1}^{\vec{z}_2} \sigma(\vec{l}, 1) d\vec{l}} = \left(T_1 |\vec{z}_1 - \vec{z}_2|^2 \right)^{\frac{\sigma^0(\epsilon_0)}{\sigma^0(1)}}.$$

The energy dependence of the cross section $\sigma^0(\epsilon_0)$ is a scalar function, which can be pre-computed and stored in a texture. Fetching a value from this texture, the compensation exponent can be easily calculated.

6. High-dimensional quadrature computation

In the previous section we concluded that the scattered contribution is a sequence of increasing dimensional integrals. Numerical quadratures generate M discrete samples $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_M$ in the domain of the integration and approximate the integral as:

$$\int f(\mathbf{s}) d\mathbf{s} \approx \frac{1}{M} \sum_{j=1}^M \frac{f(\mathbf{s}_j)}{p(\mathbf{s}_j)} \quad (8)$$

where $p(\mathbf{s}_j)$ is a density of samples. The error of the quadrature depends on two factors:

- *Importance sampling*: how well is density p proportional to original integrand f .
- *Stratification*: how accurately are sample points $\mathbf{s}_1, \mathbf{s}_2, \dots, \mathbf{s}_M$ distributed with density p .

In the integral $\tilde{y}_L^{(S)}$ of equation (6), a sample \mathbf{s}_j is a photon path connecting two detectors via S scattering points and containing an emission point somewhere:

$$\mathbf{s}_j = (\vec{s}_0^{(j)}, \vec{s}_1^{(j)}, \dots, \vec{s}_{S+1}^{(j)}) \quad \text{where } \vec{s}_0^{(j)} = \vec{z}_1 \text{ and } \vec{s}_{S+1}^{(j)} = \vec{z}_2.$$

For example, if $S = 1$ i.e. we consider single scattering, then

$$\mathbf{s}_j = (\vec{z}_1, \vec{s}^{(j)}, \vec{z}_2).$$

As the computation of a single segment of such a path requires ray-marching and therefore is rather costly, we reuse the segments of a path in many other path samples.

The basic steps of the path sampling process are shown by Fig. 5. Let us consider these steps one-by-one, examining their time complexity as well:

1. First, $N_{scatter}$ scattering points $\vec{s}_1, \dots, \vec{s}_{N_{scatter}}$ are sampled.

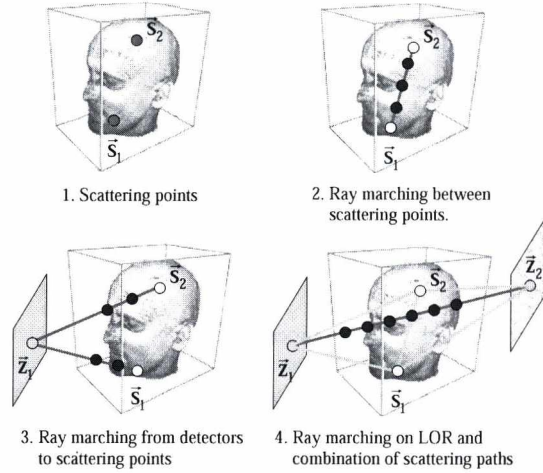


Figure 5: The sampling process consists of four steps. In step 1 the scattering points are generated by the CPU. In step 2 the GPU processes global scattering paths connecting scattering points by line segments. In step 3 a detector is connected to all scattering points by a GPU thread. In step 4 a GPU thread computes the value of the LOR including the direct and the scattered contributions.

2. In the second step global paths are generated. If we decide to simulate paths of at most S scattering points, N_{path} ordered subsets of the scattering points are selected and paths of S points are established. If statistically independent random variables were used to sample the scattering points, then the first path may be formed by points $\vec{s}_1, \dots, \vec{s}_S$, the second by $\vec{s}_{S+1}, \dots, \vec{s}_{2S}$, etc. If the sampling variables are correlated or deterministic, then they should be randomly permuted first to eliminate dependence⁵. Each path contains $S - 1$ line segments, which are marched assuming that the photon energy has not changed from the original electron energy. Note that building a path of length S , we also obtain many shorter paths as well. A path of length S can be considered as two different paths of length $S - 1$ where one of the end points is removed. Taking another example, we get $S - 1$ number of paths of length 1. Concerning the cost, rays should be marched only once, so the second step altogether marches on $N_{path}(S - 1)$ rays.
3. In the third step, each detector is connected to each of the scattering points in a deterministic manner. Each detector is assigned to a computation thread, which marches along the connection rays. The total rays processed by the third step is $N_{det}N_{scatter}$.
4. Finally, detector pairs are given to computational threads that compute the direct contribution and combine the scattering paths ending up in them. The direct contribu-

tion needs altogether $N_{detline}N_{LOR}$ ray-marching computations.

The computation effort can be analyzed by counting the number of rays needed to march on, which is

$$N_{ray} = N_{path}(S - 1) + N_{det}N_{scatter} + N_{detline}N_{LOR}.$$

In our particular case $N_{LOR} = 18 \cdot 32^4$, $S = 2$, $N_{det} = 12 \cdot 32^2$, $N_{scatter} = 128$, $N_{path} = 512$, and $N_{detline} = 16$, thus — thanks to the heavy reuse of rays — scatter compensation requires just slightly more rays than the computation of the unscattered contribution.

The described sampling process generates point samples. As these point samples are connected to all detectors, paths of length 2 (single scattering, $S = 1$) can be obtained from them. Paths longer than 2, i.e. simulating at least double scattering requires the formation of global paths. The integral quadrature of equation (8) is evaluated with these samples.

6.1. Importance sampling

Considering importance sampling, we should find a density p that mimics the integrand. When inspecting the integrand, we should take into account that we evaluate a set of integrals (i.e. an integral for every LOR) using the same set of global samples, so the density should mimic the common factors of all these integrals. The common factor is the electron density $C(\vec{v})$ of the scattering points, so we mimic this function when sampling points. We store the scattering cross section at the energy level of the electron, $\sigma(\vec{v}, 1)$, which is proportional to the electron density. As the electron density function is provided by the CT reconstruction as a voxel grid, we, in fact, sample voxels. The probability density of sampling point \vec{v} is:

$$p(\vec{v}) = \frac{\sigma(\vec{v}, 1)}{\int_V \sigma(\vec{v}, 1) dV} = \frac{\sigma_V N_{voxel}}{C \mathcal{V}},$$

where σ_V is the scattering cross section at the energy level of the electron in voxel V , $C = \sum_{i=1}^{N_{voxel}} \sigma_V$ is the sum of all voxels, and \mathcal{V} is the volume of interest.

For example, the single scattered contribution estimation with this density is the following:

$$\bar{y}_L^{(1)} \approx \frac{D_1 D_2}{N_{scatter} N_{voxel}} \cdot \sum_{j=1}^{N_{scatter}} P_{KN}(\cos \theta_j, 1) \mathcal{P}_j,$$

where θ_j is the scattering angle at \vec{s}_j , ϵ_0 is the energy level of the photon after this Compton scattering if originally it had the energy of the electron, and

$$\mathcal{P}_j = D_1^\perp(\vec{z}_1) X(\vec{z}_1, \vec{s}_j) T_1(\vec{z}_1, \vec{s}_j) I_{\epsilon_0}(\vec{s}_j, \vec{z}_2) D_{\epsilon_0}^\perp(\vec{z}_2) +$$

$$D_{\epsilon_0}^\perp(\vec{z}_1) I_{\epsilon_0}(\vec{z}_1, \vec{s}_j) X(\vec{s}_j, \vec{z}_2) T_1(\vec{s}_j, \vec{z}_2) D_1^\perp(\vec{z}_2)$$

is the total emission weighted by the attenuation of path $\vec{z}_1, \vec{s}_j, \vec{z}_2$.

6.2. Stratification

Monte Carlo methods use statistically independent random samples generated with probability density p . According to the fundamental law of statistics, the empirical distribution of independent random samples converges to the cumulative distribution obtained from p , thus Monte Carlo methods meet the stratification requirement only for larger number of samples. If we use just smaller number of samples, the distance of the empirical distribution of samples and the desired distribution may be significant, which, together with the poor mimicking of the integrand by density p , result in unacceptable error levels. As in our case the dimension of the integration domain is not very high — single scattering needs a 3D integral, the less important double scattering a 6D integral — deterministic quadratures can outperform Monte Carlo methods.

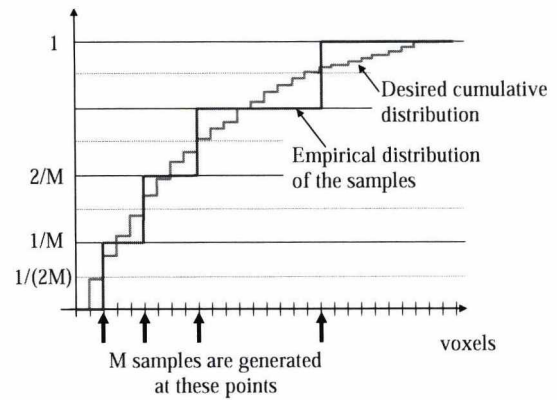


Figure 6: Sampling with delta-sigma modulation. Whenever the difference of the desired cumulative distribution and the empirical distribution of the samples get larger than $1/(2M)$, a sample is generated.

Our proposed quadrature uses Delta-Sigma modulation, which is based on the recognition that sampling with density p is in fact a *frequency modulation* problem, where the input is $p = C_V/C$ and the output is a sequence of discrete samples with density or frequency proportional to the input¹⁵. We add the ratios C_V/C one-by-one, which means that we generate the desired cumulative distribution function of the sampling strategy. If we wish to obtain M samples where M is significantly less than the number of voxels, then the empirical distribution of the real samples will differ from the desired distribution. According to the stratification requirement, the distance of the two distributions should be minimized. As the empirical distribution makes discrete jumps of size $1/M$, the error between the two distributions cannot be forced below the threshold of $1/(2M)$ (Fig. 6). However, this limit can be reached if we keep calculating the difference of the desired and empirical cumulative distributions as vox-

els are visited, and if this difference got larger than $1/(2M)$, then a new sample is generated. Compared to sampling with independent random numbers, this process generates more stratified samples. The quality of samples can be further increased if the path of visiting the voxels reflect their distances. In our current implementation, the voxel array is processed row-by-row following a serpentine path, then plane-by-plane alternating the directions. We expect even better results with Hilbert or Peano space filling curves.

The method discussed so far is strictly deterministic. If we have to randomize the samples, then the running sum of the voxel values should be initialized to a random value in $(-1/(2M), 1/(2M))$.

7. Results

The presented algorithm have been implemented in CUDA and run on nVidia GeForce 285 GFX graphics hardware. We assumed a detector system consisting of twelve square detector modules organized into a ring, and the system measures LORs connecting a detector to three other detectors being at the opposite side of the ring. A detector module consists of 32×32 detectors (left of Fig. 7). The total number of LORs is $18 \cdot (32 \times 32)^2 \approx 18 \cdot 10^6$.

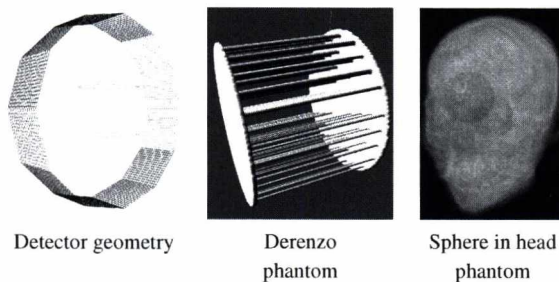


Figure 7: Geometry of the detector ring (left), the emission map of the Derenzo phantom (middle), and the sphere phantom put into the electron density of the human head (right).

We forward-projected the reference volume of a sphere (right of Fig. 7). The electron density was obtained from a CT measurement of a human head. We scaled the electron densities to guarantee that the scattered component is significant, i.e. it is 40% of the total contribution. We compared different options of the forward-projection like computing only the geometry factors and the attenuation, including single scattering, and finally adding also double scattering. To compute single and double scattering, 128 scattering points were sampled with sigma-delta modulation. According to the concept of *stochastic iteration*¹², we obtained a new set of scattering points in each iteration step. The direct contribution was calculated with 8 lines per LOR. With this number of scattering points, a single forward-projection required

13 seconds with single scattering and 20 seconds with double scattering simulation.

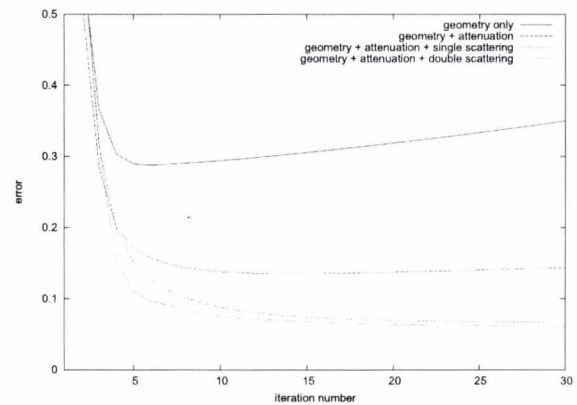


Figure 9: Effect of attenuation and single scattering compensation. A sphere inside a head phantom has been reconstructed with considering only the geometry in forward-projection, with also attenuation computation, and finally adding the contribution of single and double scattering.

The errors of the different options with respect to the iteration number are shown in Fig. 9. Note that getting the forward-projection to simulate more of the underlying physical process, the reconstruction can be made more accurate. The improvement is significant with single scattering simulation. However, the simulation of higher order scattering events helps just a little (Fig. 8).

8. Conclusion

This paper proposed a GPU based algorithm for the reconstruction of PET measurements. The original approach is restructured to exploit the massively parallel nature of GPUs. Based on the recognitions that the requirements of the GPU prefer a detector oriented viewpoint, we solve the adjoint problem during forward-projection, i.e. originate photon paths in the detectors. Making the back-projection also of gathering type, we assign voxels to parallel computational threads. The detector oriented viewpoint also allows us to reuse samples, that is, we compute many annihilation events with tracing a few line segments. The resulting approach can reduce the computation time of the fully 3D PET reconstruction to a few minutes.

Acknowledgement

This work has been supported by the Teratomo project of the NKTH, by OTKA K-719922 (Hungary), and by the Croatian-Hungarian Action Fund.

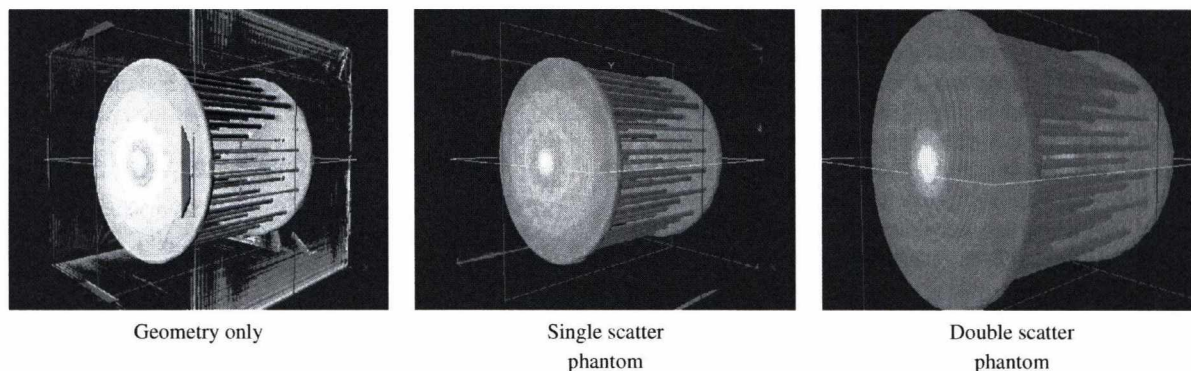


Figure 8: Reconstruction results of the Derenzo phantom.

References

- Bing Bai and Anne Smith. Fast 3D iterative reconstruction of PET images using PC graphics hardware. In *IEEE Nuclear Science Symposium*, pages 2787–2790, 2006.
- Geant. Physics reference manual, Geant4 9.1. Technical report, CERN, 2007.
- C. A. Johnson, J. Seidel, R. E. Carson, W. R. Gandler, A. Sofer, M. V. Green, and M. E. Daube-Witherspoon. Evaluation of 3D reconstruction algorithms for a small animal PET camera. *IEEE Transactions on Nuclear Science*, 44:1303–1308, June 1997.
- M. Kachelries, M. Knaup, and O. Bockenbach. Hyperfast parallel-beam and cone-beam backprojection using the CELL general purpose hardware. *Medical Physics*, 34(4):1474–1486, 2007.
- T. Kollig and A. Keller. Efficient multidimensional sampling. *Computer Graphics Forum*, 21(3):557–563, 2002.
- Miriam Leeser, Srdjan Coric, Eric Miller, Haiqian Yu, and Marc Trepanier. Parallel-beam backprojection: an FPGA implementation optimized for medical imaging. In *Proc. Tenth Int. Symposium on FPGA*, pages 217–226, 2002.
- Michel Desvignes Nicolas Gac, Stéphane Mancini and Dominique Houzet. High speed 3D tomography on CPU, GPU, and FPGA. *EURASIP Journal on Embedded Systems*, 2008. Article ID 930250.
- NVIDIA. <http://developer.nvidia.com/cuda>. In *The CUDA Homepage*, 2007.
- Guillem Prax, Carry Chinn, Peter D. Olcott, and Craig S. Levin. Fast, accurate and shift-varying line projections for iterative reconstruction using the GPU. *IEEE Trans. on Medical Imaging*, 28(3):435–445, 2009.
- D.W. Shattuck, J. Rapela, E. Asma, A. Chatzioannou, and R.M. Leahy J. Qi. Internet2-based 3D PET image reconstruction using a PC cluster. *Physics in Medicine and Biology*, 47:2785–2795, 2002.
- L. Shepp and Y. Vardi. Maximum likelihood reconstruction for emission tomography. *IEEE Trans. Med. Imaging*, 1:113–122, 1982.
- L. Szirmay-Kalos. Stochastic iteration for non-diffuse global illumination. *Computer Graphics Forum*, 18(3):233–244, 1999.
- L. Szirmay-Kalos. *Monte-Carlo Methods in Global Illumination — Photo-realistic Rendering with Randomization*. VDM, Verlag Dr. Müller, Saarbrücken, 2008.
- L. Szirmay-Kalos, L. Szécsi, and M. Sbert. *GPU-Based Techniques for Global Illumination Effects*. Morgan and Claypool Publishers, San Rafael, USA, 2008.
- László Szirmay-Kalos and László Szécsi. Deterministic importance sampling with error diffusion. *Computer Graphics Forum (EG Symposium on Rendering)*, 28(4):1056–1064, 2009.
- Zigang Wang, Guoping Han, Tianfang Li, and Zhengrong Liang. Speedup OS-EM image reconstruction by PC graphics card technologies for quantitative SPECT with varying focal-length fan-beam collimation. *IEEE Trans Nucl Sci.*, 52(5):1274–1280, 2005.
- C.C. Watson, D. Newport, and M.E. Casey. *Three-Dimensional Image Reconstruction in Radiation and Nuclear Medicine*, chapter A single scattering simulation technique for scatter correction in 3D PET, pages 255–268. Kluwer Academic Publishers, 1996.
- F. Xu and K. Mueller. GPU-acceleration of attenuation and scattering compensation in emission computed tomography. In *9th Fully Three-Dimensional Image Reconstruction in Radiology and Nuclear Medicine '07*, 2007.
- F. Xu and K. Mueller. Real-time 3D computed tomographic reconstruction using commodity graphics hardware. *Phys Med Biol*, pages 3405–3419, 2007.
- C. N. Yang. The Klein-Nishina formula & quantum electrodynamics. *Lect. Notes Phys.*, 746:393–397, 2008.

Volumetric Ambient Occlusion for Volumetric Models

Tamás Umenhoffer and László Szirmay-Kalos

BME IIT

Abstract

This paper presents a new GPU-based algorithm to compute ambient occlusion for scalar field isosurfaces. We first examine how ambient occlusion is related to the physically founded rendering equation. The correspondence is made by introducing a fuzzy membership function that defines what “near occlusions” mean. Then we develop a method to calculate ambient occlusion in real-time without any pre-computation. The proposed algorithm is based on a novel interpretation of ambient occlusion that measures how big portion of the tangent sphere of the surface belongs to the set of occluded points. The integrand of the new formula has low variation, thus can be estimated accurately with a few samples.

1. Introduction

This paper focuses on the fast computation of the reflection of the ambient light[†]. We shall assume that the primary source of illumination in the scene is a homogeneous sky light source of radiance L^a . For the sake of simplicity, we consider only diffuse surfaces. According to the rendering equation, *reflected radiance* L^r in *shaded point* \vec{x} can be obtained as:

$$L^r(\vec{x}) = \frac{a(\vec{x})}{\pi} \int_{\Omega} L^{in}(\vec{x}, \vec{\omega}) \cos^+ \theta d\omega, \quad (1)$$

where Ω is the set of directions in the hemisphere above \vec{x} , $L^{in}(\vec{x}, \vec{\omega})$ is the *incident radiance* from direction $\vec{\omega}$, $a(\vec{x})$ is the *albedo* of the surface, and θ is the angle between the surface normal and illumination direction $\vec{\omega}$. If incident angle θ is greater than 90 degrees, then the negative cosine value should be replaced by zero, which is indicated by superscript ⁺ in \cos^+ .

If no surface is seen from \vec{x} at direction $\vec{\omega}$, then shaded point \vec{x} is said to be *open* in this direction, and incident radiance L^{in} is equal to ambient radiance L^a . If there is an occluder nearby, then the point is called *closed* at this direction and the incident radiance is the radiance of the occluder surface. The exact determination of this radiance would require

a global illumination solution, which is too costly in real-time applications. Thus, we simply assume that the radiance is proportional to the ambient radiance and to a factor expressing the *openness* — also called *accessibility* — of the point. The theory of classical ambient occlusion considers an occluder to be “nearby” if its distance is smaller than a predefined threshold R . However, such an uncertain property like “being close” is better to handle by a *fuzzy measure* $\mu(d(\vec{\omega}))$ that defines how strongly direction $\vec{\omega}$ belongs to the set of open directions based on distance d of the occlusion at direction $\vec{\omega}$. In other words, this fuzzy measure expresses how an occluder at distance d allows the ambient lighting to take into effect. Relying on the physics analogy of the non-scattering participating media, a possible fuzzy measure could be $\mu(d) = 1 - \exp(-\tau d)$ where τ is the *absorption coefficient* of the media^{10,1}. Unfortunately, this interpretation requires the distance of far occlusions as well, while the classical ambient occlusion does not have to compute occlusions that are farther than R , localizing and thus simplifying the shading process.

Thus, for practical fuzzy measures we use functions that are non-negative, monotonously increasing from zero, and reach 1 at distance R . The particular value of R can be set by the application developer. When we increase this value, shadows due to ambient occlusions get larger and softer.

Using the fuzzy measure of openness, the incident radiance is

$$L^{in}(\vec{x}, \vec{\omega}) = L^a \mu(d(\vec{\omega})),$$

[†] Should the scene contain other sources, e.g. directional or point lights, their effects can be added to the illumination of the ambient light.

thus the reflected radiance (Equation 1) can be written in the following form:

$$L^r(\vec{x}) = a(\vec{x}) \cdot L^a \cdot O(\vec{x}),$$

where

$$O(\vec{x}) = \frac{1}{\pi} \int_{\Omega} \mu(d(\vec{\omega})) \cos^+ \theta d\omega. \quad (2)$$

is the *ambient occlusion* representing the local geometry. It expresses how strongly the ambient lighting can take effect on point \vec{x} . The computation of ambient occlusion requires the distances $d(\vec{\omega})$ of occluders in different directions, which are usually obtained by computationally expensive ray-tracing.

This paper proposes an ambient occlusion algorithm where the directional integral of equation 2 is replaced by a volumetric one that can efficiently be evaluated. The resulting method

- runs at high frame rates on current GPUs,
- does not require any pre-processing and thus can be applied to general dynamic models,
- can provide smooth shading with just a few samples due to partial analytic integration and interleaved sampling.

These properties make the algorithm suitable for real-time rendering and games.

2. Previous work

The oldest ambient lighting model assumes that incident radiance L^i is equal to constant L^a in all points and directions, and a point reflects $k_a L^a$ intensity, where k_a is the ambient reflectivity of the surface. As this model ignores the geometry of the scene, the resulting images are plain and do not have a 3D appearance. A physically correct approach would be the solution of the rendering equation that can take into account all factors missing in the classical ambient lighting model. However, this approach is too expensive computationally when dynamic scenes need to be rendered in real-time.

Instead of working with the rendering equation, local approaches examine only a neighborhood of the shaded point. *Ambient occlusion (AO)*^{12, 16, 11} and *obscurances*^{22, 10} methods compute just how “open” the scene is in the neighborhood of a point, and scale the ambient light accordingly. Originally, the neighborhood had a sharp boundary in ambient occlusion, and a fuzzy boundary in the obscurances method, but nowadays these terms refer to similar techniques. As the name of ambient occlusion became more popular, we also use this term in this paper.

If not only the openness of the points but also the average of open directions is obtained, then this extra directional information can be used to extend ambient occlusion from constant ambient light to environment maps¹⁶. In the

spectral extensions the average spectral reflectivities of the neighborhood and the whole scene are also taken into account, thus even *color bleeding* effects can be cheaply simulated^{13, 2}.

Since ambient occlusion is the “integrated local invisibility of the sky”, real-time methods rely on scene representations where the visibility can be easily determined. These scene representations include the approximation of surfaces by disks^{2, 9} or spheres²¹. Instead of dealing directly with the geometry, the visibility function can also be approximated³. A cube map or a depth map^{14, 20} rendered from the camera can also be considered as a sampled representation of the scene. Since these maps are already in the texture memory of the GPU, a fragment shader program can check the visibility for many directions. The method called *screen-space ambient occlusion*¹⁴ took the difference of the depth values. *Horizon split ambient occlusion*²⁰ generated and evaluated a horizon map on the fly.

When volumetric models need to be rendered, we have to find an appropriate optics analogy, for which not only the local illumination model but also non-photorealistic rendering approaches⁴ and ambient occlusion or obscurances^{17, 8, 5, 19, 18} can be applied.

3. Volumetric ambient occlusion

The evaluation of the directional integral in the ambient occlusion formula (Equation 2) requires rays to be traced in many directions, which is rather costly and needs complex GPU shaders. Thus, we transform this directional integral to a volumetric one that can be efficiently evaluated on the GPU. The integral transformation involves the following steps:

1. Considering its derivative instead of the fuzzy membership function, we replace the expensive ray tracing operation by a simple containment test. This replacement is valid if neighborhood R is small enough to allow the assumption that a ray intersects the surface at most once in interval $[0, R]$.
2. Factor $\cos^+ \theta$ in Equation 2 is compensated by domain transformation mapping the hemisphere above the surface to the tangent sphere. This makes ambient occlusion depend on the volume of that the portion of the tangent sphere which belongs to the space not occupied by the objects.

3.1. Replacing ray tracing by containment tests

Let us assume that the surfaces subdivide the space into an *outer part* where the camera is and into *inner parts* that cannot be reached from the camera without crossing the surface. We define the *characteristic function* $\mathcal{I}(\vec{p})$ that is 1 if point \vec{p} is an outer point and zero otherwise. The boundary between the inner and outer parts, i.e. the surfaces, belong to the outer part by definition (Figure 1).

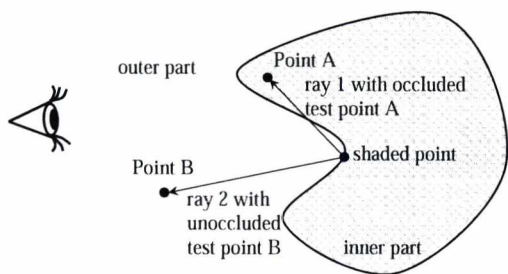


Figure 1: Replacing ray tracing by containment tests. If a test point (Point B) along a ray starting at the shaded point being in the outer region or on the region boundary is also in the outer region, then the ray either has not intersected the surface or has intersected at least two times. Supposing that the ray is short enough and thus may intersect the surface at most once, the condition of being in the outer region is equivalent to the condition that no intersection happened.

The form of the indicator function $\mathcal{I}(\vec{p})$ depends on the type or representation of the surfaces. In this paper we assume that the surface is the isosurface of a scalar field, i.e. it is defined by equation $v(\vec{p}) = v_{\vec{x}}$ where $v_{\vec{x}}$ is the scalar value of shaded point \vec{x} . In this case, the indicator function specifies whether $v \leq v_{\vec{x}}$:

$$\mathcal{I}(\vec{p}) = \epsilon(v_{\vec{x}} - v(\vec{p}))$$

where $\epsilon(x)$ is the step function, which is 1 if $x \geq 0$ and zero otherwise. In order to evaluate the ambient occlusion using containment tests, we express it as a three dimensional integral. For a point at distance d , fuzzy measure $\mu(d)$ can be found by integrating its derivative from 0 to d since $\mu(0) = 0$. Then the integration domain can be extended from d to R by multiplying the integrand by a step function which replaces the derivative by zero when the distance is greater than d :

$$\mu(d) = \int_0^d \frac{d\mu(r)}{dr} dr = \int_0^R \frac{d\mu(r)}{dr} \epsilon(d-r) dr,$$

Substituting this integral into the ambient occlusion formula we get

$$O(\vec{x}) = \frac{1}{\pi} \int_{\Omega} \int_0^R \frac{d\mu(r)}{dr} \epsilon(d-r) \cos^+ \theta dr d\omega. \quad (3)$$

Let us consider a ray of equation $\vec{x} + \vec{\omega}r$ where shaded point \vec{x} is the origin, $\vec{\omega}$ is the direction, and distance r is the ray parameter. Indicator $\epsilon(d-r)$ is 1 if the distance of the intersection d is larger than current distance r and zero otherwise, that is, it shows whether or not intersection has happened. If we assume that the ray intersects the surface at most once in the R -neighborhood, then the condition that $\vec{x} + \vec{\omega}r$ is in the outer part, i.e. $\mathcal{I}(\vec{x} + \vec{\omega}r) = 1$, also shows that no intersection

has happened yet. Thus, provided that only one intersection is possible in the R -neighborhood, indicators $\epsilon(d-r)$ and $\mathcal{I}(\vec{x} + \vec{\omega}r)$ are equivalent. Replacing step function $\epsilon(d-r)$ by indicator function \mathcal{I} in Equation 3, we obtain

$$O(\vec{x}) = \frac{1}{\pi} \int_{\Omega} \int_0^R \frac{d\mu(r)}{dr} \mathcal{I}(\vec{x} + \vec{\omega}r) \cos^+ \theta dr d\omega. \quad (4)$$

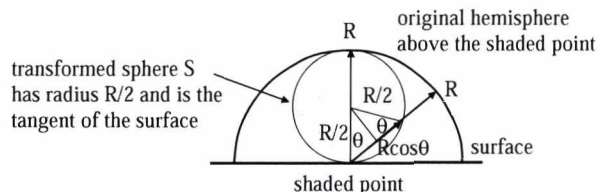


Figure 2: Transforming a hemisphere of radius R by shrinking distances by $\cos \theta$ results in another sphere of radius $R/2$.

Inspecting Equation 4 we can observe that the ambient occlusion is a double integral inside a hemisphere where the integrand includes factor $\cos^+ \theta$, thus directions enclosing a larger angle with the surface normal are less important. Instead of the multiplication, the effect of this cosine factor can also be mimicked by reducing the size of the integration domain proportionally to $\cos^+ \theta$. Note that this is equivalent to the original integral only if the other factors of the integrand are constant, and can be accepted as an approximation in other cases:

$$O(\vec{x}) \approx \frac{1}{\pi} \int_{\Omega} \int_0^{R \cos^+ \theta} \frac{d\mu(r)}{dr} \mathcal{I}(\vec{x} + \vec{\omega}r) dr d\omega.$$

Transforming a hemisphere by shrinking distances in directions enclosing angle θ with the surface normal by $\cos \theta$ results in another sphere, which is denoted by S (Figure 2). This new sphere has radius $R/2$ and its center is at distance $R/2$ from shaded point \vec{x} in the direction of the surface normal. While the original hemisphere had the shaded point as the center of its base circle, the surface will be the tangent of the new sphere at shaded point \vec{x} .

In order to replace integrals over directions and distances by a volumetric integral, let us examine the volume swept when distance r changes by dr and direction $\vec{\omega}$ varies in solid angle $d\omega$ (Figure 3). During this, we sweep a differential volume $dp = r^2 dr d\omega$. Thus, we can express the ambient occlusion as a volumetric integral in S instead of a double integral of directions and distances in the following way:

$$O(\vec{x}) \approx \frac{1}{\pi} \int_{\vec{p} \in S} \frac{d\mu(r(\vec{p}))}{dr} \frac{1}{(r(\vec{p}))^2} \mathcal{I}(\vec{p}) dp$$

where $r(\vec{p})$ is the distance of point \vec{p} from the shaded point.

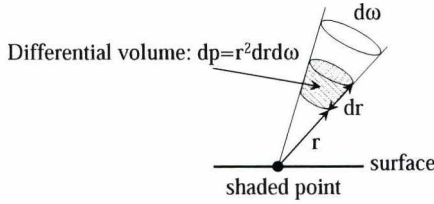


Figure 3: Differential volume swept when r changes by dr and direction $\hat{\omega}$ is in $d\omega$.

If we set the fuzzy membership function such that $d\mu(r)/dr$ is proportional to r^2 , then the ambient occlusion integral becomes just the volumetric integral of the membership function. This observation leads us to a new definition of the openness of a point, which we call the *volumetric ambient occlusion*. The volumetric ambient occlusion is the relative volume of the unoccluded part of the tangent sphere S . Formally, the volumetric ambient occlusion function is defined as:

$$O(\vec{x}) = \frac{\int_S \mathcal{I}(\vec{p}) dp}{|S|}, \quad (5)$$

where $|S| = 4(R/2)^3\pi/3$ is the volume of the tangent sphere, which makes sure that the volumetric ambient occlusion is also in $[0, 1]$.

4. Statistical evaluation of the volumetric AO

The ratio of the volumes of the unoccluded points of the tangent sphere and the volume of the tangent sphere is the geometric probability that a uniformly distributed point \vec{p} in the tangent sphere belongs to the unoccluded region, i.e. its scalar value $v(\vec{p})$ is smaller than the scalar value $v_{\vec{x}}$ of the shaded point \vec{x} or the isosurface. If the scalar field is defined by a voxel grid, then we have exact values at voxel centers, but have no information about the function $v(\vec{p})$ in between the voxel centers. Note that the usual assumption on tri-linearly varying function is just an approximation. Instead, we can also assume that the scalar value $v(\vec{p})$ is a random variable whose distribution is defined by the N voxel values v_1, \dots, v_N included in the tangent sphere. With these, the geometric probability of the occlusion is

$$O(\vec{x}) = P(v \leq v_{\vec{x}}) = \frac{\sum_{i=1}^N \mathcal{E}(v_{\vec{x}} - v_i)}{N}.$$

Note that $P(v \leq v_{\vec{x}}) = F(v_{\vec{x}})$ is the cumulative probability distribution of random variable v for value $v_{\vec{x}}$ (Figure 4).

Unfortunately, the application of this formula requires N fetches from the voxel array, which is time consuming unless N is small. However, selecting a small tangent sphere that contains just a few voxel centers degrades the quality of ambient occlusion and replaces nice soft shadows by

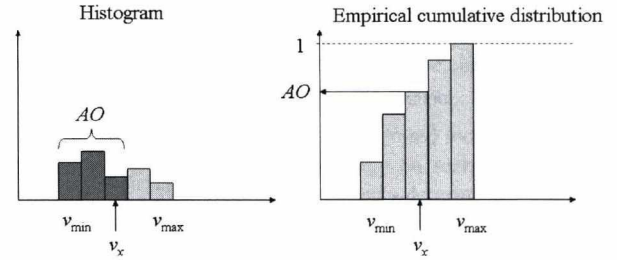


Figure 4: Ambient occlusion as the cumulative distribution function.

smaller hard ones. On the other hand, when the set of voxels contained by the tangent sphere changes, there is an abrupt change in the ambient occlusion values, which results in stripe artifacts. So it is highly desirable to find a continuous approximation of the probability distribution, which is defined by just a few representative variables⁷. Keeping real-time rendering in mind even for dynamically evolving scalar fields, we should find these parameters to allow their computation with separable filtering, so the computation cost of a single representative variable is just $\mathcal{O}(N^{1/3})$ instead of $\mathcal{O}(N)$. To make separable filtering possible, we extend the domain from the tangent sphere to its bounding box.

Examples of such representative parameters are the minimum v_{\min} , the maximum v_{\max} , and the moments $E[v^m]$, for example, the mean $E[v]$ or the second moment $E[v^2]$.

4.1. Variance volumetric AO

*Variance shadow maps*⁶ uses the mean and the second moment to estimate the geometric probability of occlusion. The required probability $P(v \leq v_{\vec{x}}) = F(v_{\vec{x}})$ for a particular point of scalar value $v_{\vec{x}}$ can be approximated using the *one tailed version of the Chebychev's inequality*, which is sharper than the classic Chebychev's inequality. For the case of $v_{\vec{x}} > E[v]$, it states that

$$1 - F(v_{\vec{x}}) = P(v > v_{\vec{x}}) \leq \frac{\sigma^2}{\sigma^2 + (v_{\vec{x}} - E[v])^2}$$

where

$$\sigma^2 = E[v^2] - E[v]^2$$

is the variance.

If $v_{\vec{x}} \leq E[v]$, then the Chebychev's inequality cannot be used, but the point is assumed to be not occluded. Variance AO reads values $E[Z]$, $E[Z^2]$ from the filtered two channel depth map, and replaces the comparison by the upper bound of the probability.

The implementation is as follows:

```

void AOShader(
    in float3 x : TEXCOORD0 // shaded point
    uniform sampler3D filVolume, // filtered
    uniform sampler3D orgVolume, // original
    uniform float R, // radius
    uniform float3 La, // ambient light
    out float col : COLOR )
{
    float vx = tex3D(orgVolume, x);
    float3 normal = Gradient(orgVolume, x);
    col = DirectLight(normal);
    float3 center = x + normal * R/2;
    float3 filt = tex3D(filVolume, center);
    float mean = filt.x;
    float secondmoment = filt.y;
    float var = secondmoment - mean * mean;
    float AO = 1; // ambient occlusion
    if (vx > mean)
        AO = 1 - var / (var + pow(vx-mean, 2));
    col += La * O;
}

```

4.2. Reconstruction of the cumulative distribution

For the sake of simplicity, we consider only the mean, but higher moments could also be handled in a similar way.

The yet unknown cumulative distribution $F(v_{\bar{x}})$ must be zero if $v_{\bar{x}} \leq v_{\min}$, equal to 1 if $v_{\bar{x}} \geq v_{\max}$, and non-decreasing in between. For notational simplicity, we introduce the following normalized parameter

$$t = \frac{v_{\bar{x}} - v_{\min}}{v_{\max} - v_{\min}} = \frac{v_{\bar{x}} - v_{\min}}{\Delta v},$$

where $\Delta v = v_{\max} - v_{\min}$.

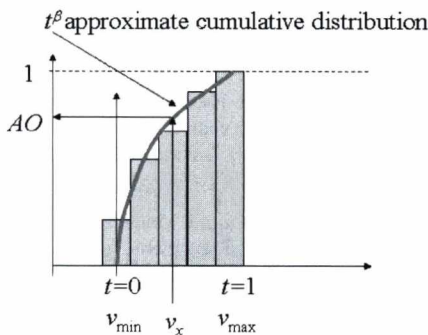


Figure 5: Approximation of the cumulative distribution function with t^β .

Using the normalized parameter, the cumulative distribution must be zero if $t \leq 0$, and 1 if $t \geq 1$, and non-decreasing in interval $[0, 1]$. We search the unknown function in the form of t^β where β is the parameter of data fitting (Figure 5). Note

that this function meets the stated requirements. Thus the cumulative distribution is

$$F(v_{\bar{x}}) = t^\beta \quad \text{where} \quad v_{\bar{x}}(t) = t\Delta v + v_{\min}.$$

Let us consider the constraint on the mean:

$$E[v] = \int_{v_{\min}}^{v_{\max}} v_{\bar{x}} dF = \int_0^1 v_{\bar{x}}(t) \frac{dF}{dt} dt = \int_0^1 (t\Delta v + v_{\min}) \beta t^{\beta-1} dt = \frac{\beta \Delta v}{\beta + 1} + v_{\min}.$$

Solving this equation for β , we get:

$$\beta = \frac{E[v] - v_{\min}}{v_{\max} - E[v]}.$$

The implementation of the algorithm in a shader is as follows:

```

void AOShader(
    in float3 x : TEXCOORD0 // shaded point
    uniform sampler3D filVolume, // filtered
    uniform sampler3D orgVolume, // original
    uniform float R, // radius
    uniform float3 La, // ambient light
    out float col : COLOR )
{
    float vx = tex3D(orgVolume, x);
    float3 normal = Gradient(orgVolume, x);
    col = DirectLight(normal);
    float3 center = x + normal * R/2;
    float3 filt = tex3D(filVolume, center);
    float vmin = filt.x;
    float vmax = filt.y;
    float mean = filt.z;
    float t = (vx-vmin)/(vmax-vmin);
    float beta = (mean-vmin)/(vmax-mean);
    float AO = pow(t, beta);
    col += La * AO;
}

```

Note that we do not check whether or not t is outside the $[0, 1]$ interval, which is correct because $v_{\bar{x}}$ is included in the neighborhood of the computing the maximum and the minimum, so it can never happen that $v_{\bar{x}}$ is outside of the interval of the minimum and the maximum.

5. Opacity modulation and gamma correction

The generated images can be further improved by incorporating the ambient occlusion in the opacity transfer function or emphasizing the ambient occlusion effect by an additional gamma correction.

As Ruiz et al.¹⁹ proposed the derivative of the ambient occlusion indicates where the salient regions are, so it is worth modulating the opacity value by this derivative. They use the gradient magnitude of the ambient occlusion, which is

estimated by a 4D regression filter¹⁵. However, the gradient calculation is a non-separable filter, which has a significant computational cost. Instead of the gradient magnitude we propose the application of the variance of the scalar value, which also indicates where the volume changes quickly. The variance can be obtained as the difference of the second moment and the square of the mean value and both of them can be obtained by separable filtering. Thus modified opacity α' is:

$$\alpha' = \alpha \cdot \min\left(\frac{\sigma^2}{\sigma_{\max}^2}, 1\right) = \alpha \cdot \min\left(\frac{E[v^2] - E^2[v]}{\sigma_{\max}^2}, 1\right)$$

where σ_{\max}^2 is a user defined constant.

Finally, in order to further emphasize the ambient occlusion, we can apply contrast enhancement using gamma correction with exponent γ . It means that we use O^γ instead of ambient occlusion O in the lighting calculations.

6. Results

The proposed methods have been implemented in DirectX/HLSL and their performance has been measured on an NVIDIA GeForce 9600M GPU at 800×800 resolution. On this hardware the isosurfaces of the 128^3 resolution head model are rendered at 20 FPS, the transparent rendering runs at 10 FPS.

Figure 6 shows the isosurface of a volumetric head model rendered with the method reconstructing the cumulative distribution. We set the gamma-correction factor differently to emphasize ambient occlusion effects. Figure 7 demonstrates transparent volume rendering with variance based ambient occlusion and with the ambient occlusion based on the reconstruction of the cumulative distribution function. We used the original opacities, i.e. we have not applied the proposed modulation scheme. Finally, Figure 8 compares the method using ambient occlusion based on the reconstruction of the cumulative distribution function and also modulating the alpha value proportionally with the variance to classical transparent volume rendering. Note that the modulation of the alpha value emphasizes the interesting parts making homogeneous regions transparent.

7. Conclusions

This paper proposed a fast method for the computation of ambient occlusion in volumetric models. The method is based on the estimation of the probability that the point is occluded. The computation requires separable filtering, thus it is feasible even for dynamically evolving fields.

Acknowledgement

This work has been supported by the Teratomo project of National Office for Research and Technology, OTKA K-719922 (Hungary), and by the Hungarian-Croatian Fund.

Appendix

Here we present a relatively simple proof that is based on the *Markov inequality* stating that $P(\xi \geq a) \leq E[\xi]/a$ holds for a non negative random variable ξ and positive constant a , substituting $\xi = (v - E[v] + \sigma^2/(v_{\bar{x}} - E[v]))^2$ and $a = (v_{\bar{x}} - E[v] + \sigma^2/(v_{\bar{x}} - E[v]))^2$. Let us consider the case when $v_{\bar{x}} > E[v]$. The left side of the Markov inequality can be written as follows:

$$P(\xi \geq a) =$$

$$P\left(\left(v - E[v] + \frac{\sigma^2}{v_{\bar{x}} - E[v]}\right)^2 \geq \left(v_{\bar{x}} - E[v] + \frac{\sigma^2}{v_{\bar{x}} - E[v]}\right)^2\right) =$$

$$P\left(v - E[v] + \frac{\sigma^2}{v_{\bar{x}} - E[v]} \geq v_{\bar{x}} - E[v] + \frac{\sigma^2}{v_{\bar{x}} - E[v]}\right) = P(v \geq v_{\bar{x}}).$$

The right side of the Markov inequality is

$$\frac{E[\xi]}{a} = \frac{E\left[\left(v - E[v] + \frac{\sigma^2}{v_{\bar{x}} - E[v]}\right)^2\right]}{\left(v_{\bar{x}} - E[v] + \frac{\sigma^2}{v_{\bar{x}} - E[v]}\right)^2} =$$

$$\frac{\sigma^2 + \sigma^4/(v_{\bar{x}} - E[v])^2}{(v_{\bar{x}} - E[v] + \sigma^2/(v_{\bar{x}} - E[v]))^2} =$$

$$\sigma^2 \frac{(v_{\bar{x}} - E[v])^2 + \sigma^2}{(v_{\bar{x}} - E[v])^4 + 2\sigma^2(v_{\bar{x}} - E[v])^2 + \sigma^4} = \frac{\sigma^2}{\sigma^2 + (v_{\bar{x}} - E[v])^2}.$$

References

1. T. Annen, T. Mertens, H.-P. Seidel, E. Flerackers, and J. Kautz. Exponential shadow maps. In *GI '08: Proceedings of graphics interface 2008*, pages 155–161, Toronto, Ont., Canada, Canada, 2008. Canadian Information Processing Society.
2. M. Bunnell. Dynamic ambient occlusion and indirect lighting. In M. Parr, editor, *GPU Gems 2*, pages 223–233. Addison-Wesley, 2005.
3. P. Clarberg and T. Akenine-Möller. Exploiting Visibility Correlation in Direct Illumination. *Computer Graphics Forum (Proceedings of EGSR 2008)*, 27(4):1125–1136, 2008.
4. B. Csébfalvi, L. Mroz, H. Hauser, A. König, and E. Gröller. Fast visualization of object contours by non-photorealistic volume rendering. *Computer Graphics Forum*, 20(3):452–460, 2001.
5. J. Diaz, H. Yela, and P. Vázquez. Vicinity occlusion maps: Enhanced depth perception of volumetric models. In *Computer Graphics International*, 2008.
6. W. Donnelly and A. Lauritzen. Variance shadow maps. In *SI3D '06: Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 161–165, 2006.
7. H. Grun. Approximate cumulative distribution function shadow mapping. In Wolfgang Engel, editor, *Shader X 7*. Course Technology, 2008.
8. F. Hernel, A. Ynnerman, and P. Ljung. Efficient ambient and emissive tissue illumination using local occlusion in multiresolution volume rendering. In *Eurographics/IEEE-VGTC Symposium on Volume Graphics*, pages 1–8, Prague, 2007.

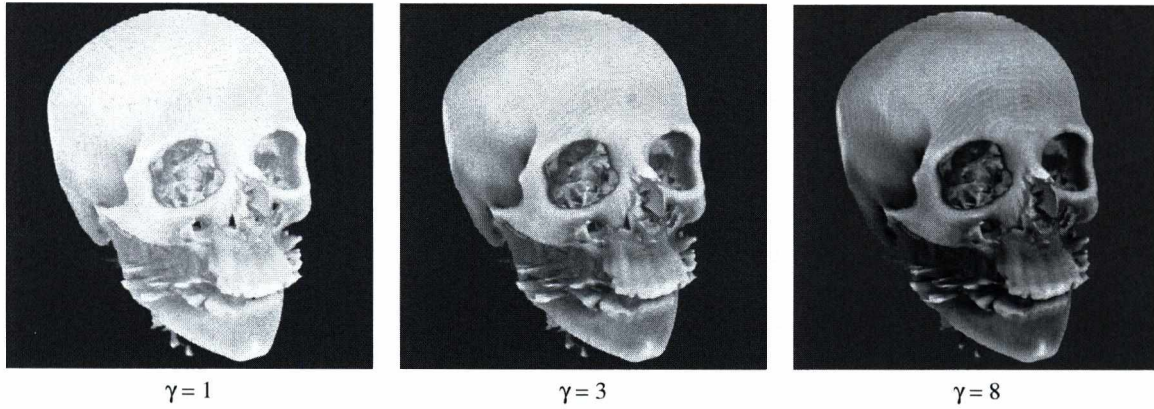


Figure 6: Isosurface rendering with the method reconstructing of the cumulative distribution taking different gamma correction values.

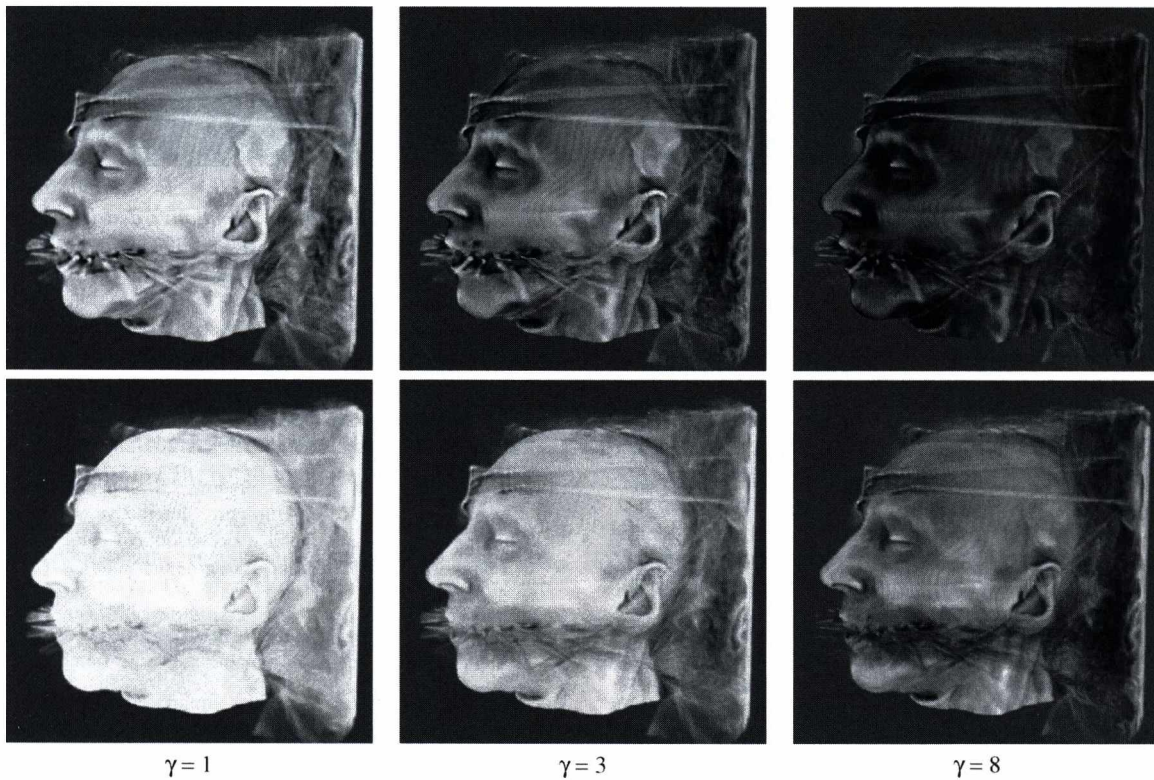


Figure 7: Transparent volume rendering where we compared the method reconstructing of the cumulative distribution (lower row) to variance ambient occlusion (upper row). We used the original opacity values in both cases and have not applied the proposed modulation.

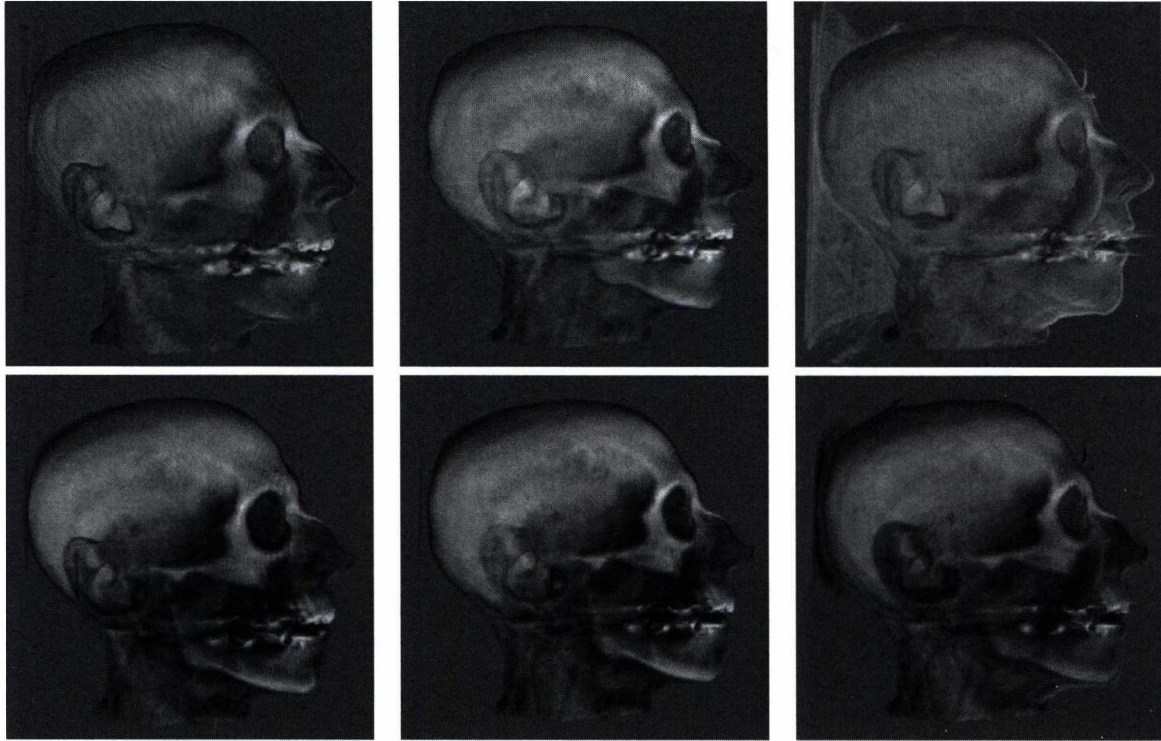


Figure 8: Transparent volume rendering where we compared the method reconstructing of the cumulative distribution with opacity modulation (lower row) to the original volume rendering algorithm using no ambient occlusion (upper row).

9. J. Hoberock and Y. Jia. High-quality ambient occlusion. In Hubert Nguyen, editor. *GPU Gems 3*, pages 257–274. Addison-Wesley, 2007.
10. A. Iones, A. Krupkin, M. Sbert, and S. Zhukov. Fast realistic lighting for video games. *IEEE Computer Graphics and Applications*, 23(3):54–64, 2003.
11. J. Kontkanen and T. Aila. Ambient occlusion for animated characters. In *Proceedings of the 2006 Eurographics Symposium on Rendering*, 2006.
12. Hayden Landis. Production-ready global illumination. Technical report, SIGGRAPH Course notes 16, 2002.
13. A. Méndez, M. Sbert, and J. Catá. Real-time obscurances with color bleeding. In *SCCG '03: Proceedings of the 19th spring conference on Computer graphics*, pages 171–176, 2003.
14. M. Mitting. Finding next gen — CryEngine 2. In *Advanced Real-Time Rendering in 3D Graphics and Games Course - Siggraph 2007*, pages 97–121. 2007.
15. L. Neumann, B. Csébfalvi, A. König, and E. Gröller. Gradient estimation in volume data using 4D linear regression. In M. Gross and F. R. A. Hopgood, editors. *Computer Graphics Forum (Eurographics 2000)*, volume 19(3), pages 351–358, 2000.
16. M. Pharr and S. Green. Ambient occlusion. In *GPU Gems*, pages 279–292. Addison-Wesley, 2004.
17. C. Rezk-Salama. Production-Ready GPU-based Monte-Carlo Volume Raycasting. Technical report, 2007.
18. T. Ropinski, J. Meyer-Spradow, S. Diepenbrock, J. Mensmann, and K. H. Hinrichs. Interactive volume rendering with dynamic ambient occlusion and color bleeding. *Computer Graphics Forum (Eurographics 2008)*, 27(2):567–576, 2008.
19. M. Ruiz, I. Boada, I. Viola, S. Bruckner, M. Feixas, and M. Sbert. Obscurance-based volume rendering framework. In *Proceedings of Volume Graphics 2008*, 2008.
20. Miguel Sainz. Real-time depth buffer based ambient occlusion. In *Games Developers Conference '08*. 2008.
21. P. Shanmugam and O. Arikan. Hardware accelerated ambient occlusion techniques on GPUs. In *Proceedings of the 2007 Symposium on Interactive 3D graphics*, pages 73–80, 2007.
22. S. Zhukov, A. Iones, and G. Kronin. An ambient light illumination model. In *Proceedings of the Eurographics Rendering Workshop*, pages 45–56, 1998.

Monte Carlo Radiative Transport on the GPU

Balázs Tóth, Milán Magdics

BME IIT

Abstract

This paper presents a fast parallel Monte Carlo method to solve the radiative transport equation in inhomogeneous participating media. The implementation is based on CUDA and runs on the GPU. In order to meet the requirements of the parallel GPU architecture and to reuse shooting paths, we follow a photon mapping approach where during gathering the radiance is computed for each voxel. During shooting we consider two different ways of sampling the free run lengths, ray marching and Woodcock tracking. We also discuss the generalization of the method to γ -photons that are relevant in medical simulation, especially in radiotherapy treatment design.

1. Introduction

The multiple-scattering simulation in participating media is one of the most challenging problems in computer graphics, radiotherapy, SPECT reconstruction, etc. In engineering and medical data analysis we need not only pleasing images, but also solutions with controllable accuracy. As these applications rely on intensive man-machine communication, the system is expected to respond to user actions interactively and deliver simulation results in seconds rather than in hours, which is typical in CPU based solutions.

This paper proposes a Monte Carlo solution to interactively render inhomogeneous participating media defined by large voxel arrays. The algorithm is developed with the aim of efficient GPU implementation¹⁹. In order to reduce the communication overhead of parallel computing nodes, we restructure classical photon tracing to make all computation threads independent and to avoid all communication and synchronization among them. The resulting algorithm is similar to volumetric photon mapping in the sense that it is decomposed to a shooting part where paths originating from the source are sampled randomly and to a deterministic gathering part that obtains the pixel radiances.

The main difference of our approach is that the shooting part prepares voxel radiances, i.e. it not only registers scattering points and powers, but also executes filtering in each voxel. This way, the gathering part is a simple back-to-front volume rendering method, which can be efficiently executed on the GPU.

The paper is organized as follows. Section 2 discusses the theory of radiative transfer both for light and γ -photons, and surveys the related previous work. Section 3 presents the new parallel algorithm. Section 4 summarizes the results.

2. Radiative transport

When photons interact with participating media, they scatter either on electrons or less probably on atomic cores. A photon has zero rest mass, but non-zero energy and impulse, so it can be associated with relativistic mass $m = E/c^2 = h\nu/c^2$ where E is the energy of the photon, h is the Planck constant, ν is the frequency of the radiation, and c is the speed of light. In case of the visible spectrum, the relativistic mass of a photon is negligible with respect to the mass of electrons or atomic cores, so when a photon elastically scatters on an electron it bounces off like hitting a rigid wall, keeping its energy and consequently its original frequency. In case of inelastic scattering, also called *photoelectric effect*, the photon's energy is absorbed. Thus, upon scattering, the number of light photons reduces but the frequency of the remaining photons does not change. This is why we can handle frequencies independently in computer graphics.

On higher energy or frequency ranges, however, photon energy and impulse become comparable to the energy and impulse of electrons. Thus, scattering may modify not only the number of photons but also their frequency, so frequencies become coupled and cannot be handled independently. This frequency range is particularly important in medical

simulation since CT, PET, SPECT, etc. devices work with γ -photons.

2.1. Light photons

Photons of visible light do not change their frequency upon scattering on the medium particles. So we can solve the transport problem independently on each of the representative frequencies, usually corresponding to red, green, and blue light. Multiple scattering simulation should solve the *radiative transport equation* that expresses the change of radiance $L(\vec{x}, \vec{\omega})$ at point \vec{x} and in direction $\vec{\omega}$:

$$\vec{\omega} \cdot \vec{\nabla} L = \frac{dL(\vec{x} + \vec{\omega}s, \vec{\omega})}{ds} \Big|_{s=0} = -\sigma_t(\vec{x})L(\vec{x}, \vec{\omega}) + \sigma_s(\vec{x}) \int_{\Omega} L(\vec{x}, \vec{\omega}') P(\vec{\omega}' \cdot \vec{\omega}) d\omega'. \quad (1)$$

In this equation the negative term represents *absorption* and *out-scattering*, i.e. when photons coming from direction $\vec{\omega}$ collide and upon collision they are absorbed or scattered in another direction (Figure 1). The probability of collision in a unit distance is defined by *extinction coefficient* σ_t , which is broken down to *scattering coefficient* σ_s and *absorption coefficient* σ_a according to the two possible events of scattering and absorption:

$$\sigma_t(\vec{x}) = \sigma_s(\vec{x}) + \sigma_a(\vec{x}).$$

The probability of reflection given that collision happened is called the *albedo* of the material:

$$a = \frac{\sigma_s}{\sigma_t}.$$

The positive term in the right side of equation (1) represents *in-scattering*, i.e. the contribution of photons coming from other directions $\vec{\omega}'$ and get scattered to direction $\vec{\omega}$. The probability that non-absorbing scattering happens in unit distance is σ_s . The probability density of the reflection direction is defined by *phase function* $P(\cos\theta)$ that depends on the cosine of the scattering angle $\cos\theta = \vec{\omega}' \cdot \vec{\omega}$. In order to consider all incident directions, the contributions should be integrated for all directions $\vec{\omega}'$ of the directional sphere Ω .

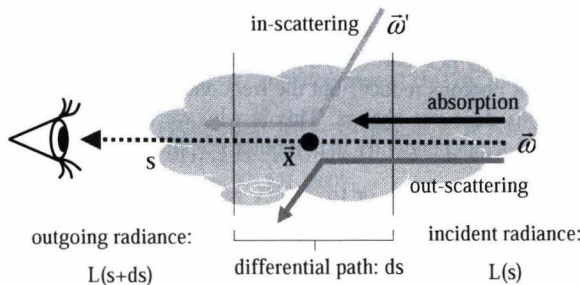


Figure 1: Change of radiance in participating media.

In *isotropic* (also called diffuse) scattering, the reflected radiance is uniform, and thus the phase function is constant:

$$P_{\text{isotropic}}(\vec{\omega}' \cdot \vec{\omega}) = \frac{1}{4\pi}.$$

For anisotropic scattering, the phase function varies with the scattering angle. The extent of anisotropy is usually expressed by the mean cosine of the scattering angle:

$$g = \int_{\Omega} (\vec{\omega}' \cdot \vec{\omega}) P(\vec{\omega}' \cdot \vec{\omega}) d\omega'.$$

An example of anisotropic scattering is the *Rayleigh scattering*, which is also responsible for sky colors. The phase function of Rayleigh scattering is:

$$P_{\text{Rayleigh}}(\vec{\omega}' \cdot \vec{\omega}) = \frac{3}{16\pi} (1 + (\vec{\omega}' \cdot \vec{\omega})^2) = \frac{3}{16\pi} (1 + \cos^2\theta).$$

2.2. γ -photons

The scattering of γ -photons is described by the *Klein-Nishina* formula ²¹, which expresses the *differential cross section*, i.e. the product of the energy dependent phase function and the scattering coefficient:

$$\sigma_s(\vec{x}, E_0) P_{KN}(\cos\theta, E_0) = C(\vec{x})(\epsilon + \epsilon^3 - \epsilon^2 \sin^2\theta),$$

where $\cos\theta = \vec{\omega} \cdot \vec{\omega}'$ is the scattering angle, $\epsilon = E_1/E_0$ expresses the ratio of the scattered energy E_1 and the incident energy E_0 , and $C(\vec{x})$ is the product of the electron density (number of electrons per unit volume) at point \vec{x} and a scattering factor $r_e^2/2$ where $r_e = 2.82 \cdot 10^{-15}$ [m] is the *classical electron radius*.

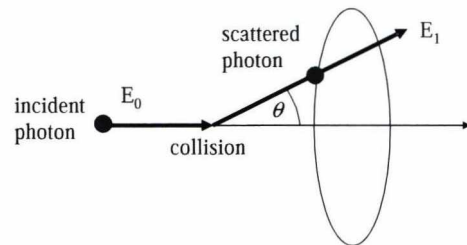


Figure 2: Compton scattering. The relative energy change E_1/E_0 is determined by the scattering angle θ .

When scattering happens, there is a unique correspondence between the relative scattered energy ϵ and the cosine of the scattering angle θ , as defined by the *Compton formula* (Figure 2):

$$E_1 = \frac{E_0}{1 + \frac{E_0}{m_e c^2} (1 - \cos\theta)},$$

where $m_e c^2$ is the electron's energy expressed by the product of its rest mass m_e and the square of the speed of light c .

From the Klein-Nishina formula, the scattering coefficient

can be obtained by the directional integration over the sphere Ω since the phase function is the probability density of the scattering direction, thus its directional integral is 1:

$$\begin{aligned}\sigma_s(\vec{x}, E_0) &= \int_{\Omega} \sigma_s(\vec{x}, E_0) P_{KN}(\cos \theta, E_0) d\omega = \\ C(\vec{x}) &\int_0^{2\pi} \int_{-1}^1 \epsilon + \epsilon^3 - \epsilon^2 \sin^2 \theta d \cos \theta d\phi = \\ C(\vec{x}) 2\pi &\int_{-1}^1 \epsilon + \epsilon^3 - \epsilon^2 (1 - \cos^2 \theta) d \cos \theta\end{aligned}$$

as $d\omega = d\phi d \cos \theta$ if θ is the angle between the original and the scattering direction and ϕ is the angle between the projection of the scattering direction onto a plane perpendicular to the original direction and a reference vector in this plane.

Note that the scattering coefficient is a product of the electron density $C(\vec{x})$ and a factor that is independent of the location of this point. This second factor would be the scattering coefficient for unit electron density material, and is denoted by

$$\sigma_s(E_0) = 2\pi \int_{-1}^1 \epsilon + \epsilon^3 - \epsilon^2 (1 - \cos^2 \theta) d \cos \theta. \quad (2)$$

With this function, the scattering coefficient and the phase function can be expressed as

$$\sigma_s(\vec{x}, E_0) = \sigma_s(E_0) C(\vec{x}), \quad (3)$$

$$P_{KN}(\cos \theta, E_0) = \frac{\epsilon + \epsilon^3 - \epsilon^2 \sin^2 \theta}{\sigma_s(E_0)}.$$

Note that in these equations the new energy (ϵ, E) is not an independent variable, it is unambiguously determined by the incident energy E_0 and the scattering angle according to the Compton formula.

Examining the Compton formula, we can conclude that the energy change at scattering is significant when E_0 is non negligible with respect to the energy of the electron. This is not the case for photons of visible wavelengths, when $E_0 \ll m_e c^2$, thus $E_1 \approx E_0$. In this case, the Klein-Nishina phase function becomes similar to the phase function of Rayleigh scattering:

$$P_{KN}(\cos \theta, E_0) \approx \frac{1}{\sigma_s(E_0)} (1 + \cos^2 \theta) = P_{Rayleigh}(\cos \theta).$$

Due to the coupling of different frequencies, we cannot consider the transport equation on different photon energy levels independently. Instead, a single equation describes the transport on all levels:

$$\left. \frac{dL(\vec{x} + \vec{\omega}s, \vec{\omega}, E_1)}{ds} \right|_{s=0} = -\sigma_t(\vec{x}, E_1) L(\vec{x}, \vec{\omega}, E_1) +$$

$$\int_{\Omega} L(\vec{x}, \vec{\omega}', E_0) \sigma_s(\vec{x}, E_0) P_{KN}(\vec{\omega}' \cdot \vec{\omega}, E_0) d\omega'. \quad (4)$$

The Compton formula unambiguously determines the original photon energy E_0 included in this formula from scattered photon energy E_1 and the cosine of the scattering angle.

2.3. Model representation

In *homogeneous media* volume properties σ_t and σ_s do not depend on position \vec{x} . In *inhomogeneous media* these properties depend on the actual position.

In case of measured data, material properties are usually stored in a 3D voxel grid, and are assumed to be constant or linear between voxel centers. Let Δ be the distance of the grid points. The total extinction of a voxel can be expressed by the following new parameter that is called the *opacity* and is denoted by α :

$$\alpha = 1 - e^{-\sigma_t \Delta} \approx \sigma_t \Delta. \quad (5)$$

The primary source of the illumination may be the surfaces, light sources, or the volume itself. These can be taken into account by either adding a source term to the right side of equation (1) or by enforcing boundary conditions making the radiance of the volume equal to the prescribed radiance of the source.

In this paper we assume that the primary source of illumination is a single point source. This case has not only high practical relevance, but more complex sources can also be traced back to the collection of point sources.

2.4. Solution methods

Cerezo et al. ² classified algorithms solving the radiative transport equation as analytic, stochastic, and iterative.

2.4.1. Analytic solution methods

Analytic techniques rely on simplifying assumptions, such that the volume is homogeneous, and usually consider only the single scattering case ^{1, 15}.

Single scattering approximation

Radiance $L(\vec{x}, \vec{\omega})$ is often expressed as the sum of two terms, the *direct term* L_d that represents unscattered light, and the *media term* L_m that stands for the light component that scattered at least once:

$$L(\vec{x}, \vec{\omega}) = L_d(\vec{x}, \vec{\omega}) + L_m(\vec{x}, \vec{\omega}).$$

The direct term is reduced by absorption and out-scattering:

$$\frac{dL_d}{ds} = -\sigma_t(\vec{x}) L_d(\vec{x}, \vec{\omega}).$$

The media term is not only reduced by absorption and out-scattering, but also increased by in-scattering:

$$\frac{dL_m}{ds} = -\sigma_t(\vec{x})L_m(\vec{x}, \vec{\omega}) + \sigma_s(\vec{x}) \int_{\Omega} (L_d(\vec{x}, \vec{\omega}) + L_m(\vec{x}, \vec{\omega}')) P(\vec{\omega}' \cdot \vec{\omega}) d\omega'$$

Note that this equation can be re-written by considering the reflection of the direct term as a *volumetric source*:

$$\frac{dL_m}{ds} = -\sigma_t(\vec{x})L_m(\vec{x}, \vec{\omega}) + \sigma_s(\vec{x}) \int_{\Omega} L_m(\vec{x}, \vec{\omega}') P(\vec{\omega}' \cdot \vec{\omega}) d\omega' + \sigma_s(\vec{x}) Q(\vec{x}, \vec{\omega}), \quad (6)$$

where the source intensity is:

$$Q(\vec{x}, \vec{\omega}) = \int_{\Omega} L_d(\vec{x}, \vec{\omega}') P(\vec{\omega}' \cdot \vec{\omega}, \vec{x}) d\omega'$$

The single scattering approximation ignores the in-scattering of the media term, and simplifies equation (6) as:

$$\frac{dL_m}{ds} = -\sigma_t(\vec{x})L_m(\vec{x}, \vec{\omega}) + \sigma_s(\vec{x})Q(\vec{x}, \vec{\omega}).$$

This is a linear differential equation for the media term, which can be solved analytically. For homogeneous medium, the integrals of the solution can be expressed in a tabular¹⁵ or even in closed analytic form¹¹. In inhomogeneous medium, *ray marching* should be executed twice: once for light rays to get Q , and once for accumulating the radiance for visibility rays.

2.5. Stochastic solution methods

In order to get the radiance of a point, all photon paths connecting the light source to this point via arbitrary number of scattering points should be considered and their contributions be added. As the location of a single scattering can be specified by three Cartesian coordinates, the contribution of paths of length l can be expressed as a $3l$ -dimensional integral. As l can be arbitrary, we should deal with high-dimensional (in fact infinite-dimensional) integrals. Such high-dimensional integrals can be evaluated by Monte Carlo quadrature that samples the high-dimensional domain randomly and approximates the integral as the average of the contribution of these random paths, divided by the probability of the samples¹⁶. The error of Monte Carlo quadrature taking N samples is in $O(N^{-1/2})$. To speed up the computation by a linear factor, ray samples are reused for many pixels, storing partial results, for example, in photon maps^{6, 12}.

2.5.1. Iterative solution methods

Iteration obtains the solution as the limiting value of an iteration sequence. In order to store temporary radiance estimates, we use a finite element representation. Popular finite

element representations include the zonal method¹³, spherical harmonics⁸, radial basis functions²², metaballs, etc. or can exploit the particle system representation¹⁸ or BCC³ and FCC grids¹⁷.

Substituting the finite element approximation, the transport equation and the projection into the finite element basis simplify to a system of linear equations. Iteration obtains the solution as the limiting value of the iteration sequence. The convergence is guaranteed if \mathbf{T} is a *contraction*, i.e. for some norm of this matrix, we have

$$\|\mathbf{T} \cdot \mathbf{L}\| < \lambda \|\mathbf{L}\|, \quad \lambda < 1,$$

which is the case if the albedo is less than 1.

Note the error is proportional to the norm of the difference of the initial guess \mathbf{L}_0 and the final solution \mathbf{L} and reduces with the speed of a geometric series, i.e. it is in $O(\lambda^N)$ after N steps.

3. The proposed method

As photons travel in the considered volume, they may get scattered several times before they get absorbed, leave the volume or get captured by a detector. The material properties are defined by a 3D voxel grid at high resolutions. The transport problem is usually solved by Monte Carlo simulation that directly mimics the physical process. However, this approach cannot reuse partial paths.

In order to re-use a single random sample for all camera positions, the path simulation is decomposed to random *shooting* part and to a deterministic *gathering* part. We also define a global *radiance estimation* part, which can be executed independently or can be merged with shooting.

During shooting, random paths are generated that originate in the source. A single path contains several rays. The output of shooting is the collection of scattering points and the incident energies.

The global radiance estimation part converts these scattering points to reflected radiance values in voxels. If m particles are scattered at voxel of volume ΔV and their powers and incident directions are $\Delta\Phi_1, \dots, \Delta\Phi_m$ and $\vec{\omega}_1, \dots, \vec{\omega}_m$, respectively, then the radiance of the voxel located at point \vec{x} is

$$L(\vec{x}, \vec{\omega}_{eye}) = \frac{1}{\sigma_t(\vec{x})} \frac{d^2\Phi}{d\omega dV} \approx \frac{1}{\sigma_t(\vec{x})} \frac{\sum_{i=1}^m \Delta\Phi_i P(\vec{\omega}_i, \vec{\omega}_{eye})}{\Delta V}$$

The merging of this step with shooting has both advantages and disadvantages. The disadvantage is that threads generating different paths must accumulate their contribution in the global 3D voxel array, which requires atomic add operations. The separate global radiance estimation could solve this problem without atomic adds if we allocate a thread for every voxel, but this extra pass has additional computational cost. Here we discuss a solution that obtains the particles

scattered in a given voxel, and immediately calculate the radiance towards the eye in the shooting part.

Finally, the gathering part obtains the pixel values for a particular camera position by tracing rays from the eye through the image pixels. As the deterministic connection does not consider additional scattering events, only the accumulated extinction needs to be calculated between the scattering points and the eye.

3.1. Shooting

Our shooting algorithm computes the radiance at every voxel, by simulating random paths. Originally, the radiance L is set to zero, then the radiance values are increased by the shooting path contributions.

```

for each path sample do
  terminated = FALSE;
  Generate initial direction  $\vec{\omega}$ ;
  while not terminated;
    Advance by free path length to voxel;
    if out of volume then terminated = TRUE;
    else
       $L_{\text{voxel}} += \Phi_{\text{photon}} P(\vec{\omega}_{\text{photon}}, \vec{\omega}_{\text{eye}})$ ;
      Absorption prob =  $\sigma_a(\text{voxel}) / \sigma_t(\text{voxel})$ ;
      if absorption then terminated = TRUE;
      else Sample new direction  $\vec{\omega}$ ;
    endif
  endwhile
endfor
for each voxel do  $L_{\text{voxel}} /= \Delta V_{\text{voxel}} \sigma_t(\text{voxel})$ ;

```

The random part of the algorithm involves free-path sampling, absorption handling, and scattering angle sampling. The deterministic part computes accumulated extinction. In the following subsections we discuss how these elementary tasks can be solved by parallel programs.

3.1.1. Free path sampling

We implemented two different free path sampling methods.

Inversion method

The cumulative probability distribution of the length along a ray of origin \vec{x} and direction $\vec{\omega}$ is

$$P(s) = 1 - \exp\left(-\int_0^s \sigma_t(\vec{x} + \vec{\omega}S) dS\right),$$

which can be sampled by transforming a uniformly distributed random variable r_1 and finding $s = n\Delta s$ where a running sum exceeds the threshold of the transformed variable:

$$\sum_{i=0}^{n-1} \sigma_t(\vec{x} + \vec{\omega}i\Delta s)\Delta s \leq -\log r_1 < \sum_{i=0}^n \sigma_t(\vec{x} + \vec{\omega}i\Delta s)\Delta s.$$

For light photons total cross section σ_t is specified independently on the representative wavelengths.

Woodcock tracking

*Woodcock tracking*²⁰ provides an alternative method to *ray marching* that visits all voxels along a ray. Woodcock tracking samples free path lengths with the maximal extinction coefficient σ_t^{\max} , advances the ray with sampled distance $-(\log r)/\sigma_t^{\max}$ where r is a uniformly distributed random variable in $(0, 1]$, and decides with probability σ_t/σ_t^{\max} whether it is a real collision point or a virtual one due to not sampling with the real extinction coefficient. If a virtual point is found, then neither the energy nor the direction is changed, and the same sampling step is repeated from there.

Let us consider the change of the cumulative distribution $P(s)$ by ds :

$$P(s + ds) = P(s) + (1 - P(s))\sigma_t(s)ds.$$

Woodcock tracking generates samples as

$$P(s + ds) = P(s) + (1 - P(s))\sigma_t^{\max} ds =$$

$$P(s) + (1 - P(s))\sigma_t(s)ds + (1 - P(s))(\sigma_t^{\max} - \sigma_t(s))ds$$

where $P(s)\sigma_t(s)ds$ and $P(s)(\sigma_t^{\max} - \sigma_t(s))ds$ are the probabilities of real and virtual collisions, respectively. Thus, if a collision is declared as real with probability $\sigma_t(s)/\sigma_t^{\max}$, then we select according to the real extinction coefficient.

3.2. Extension to γ -photons

For γ -photons we can use the same methods. The only difference is that the simulation is executed on many wavelengths, so we cannot assume that the voxel array stores the extinction parameters for the wavelength of the simulation. Instead, we store the scaled electron density $C(\vec{x})$ in the voxel array, and obtain the extinction parameters on the fly depending on the energy of the simulated photons. We shall assume that the photoelectric effect is negligible, thus the extinction coefficient becomes equal to the scattering coefficient, which is computed as the product of the electron density fetched from the voxel array and the integral of equation 2. Instead of computing this integral each time when a scattering coefficient is needed, we prepare these values in a one-dimensional texture (Figure 3) in the preprocessing phase for $E_0 = E_{\max}/128, 2E_{\max}/128, \dots, E_{\max}$ where E_{\max} is the energy of the photons emitted by the source. During sampling, $\sigma_s(\vec{x}, E_0)$ is obtained as the product of the prepared values and the scaled density of electrons C as specified in equation (3).

3.3. Scattering direction

3.3.1. Light photons

We consider the case of isotropic scattering for light photons, thus the scattering direction has uniform distribution.

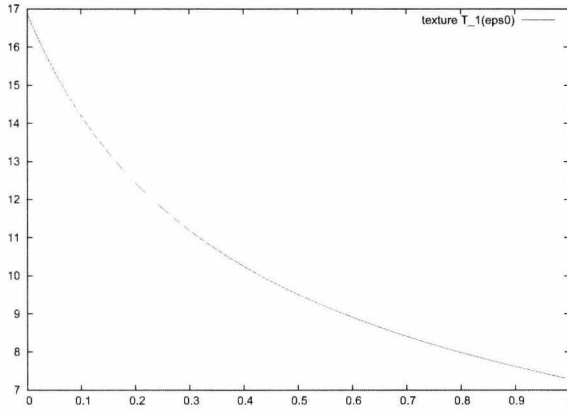


Figure 3: Texture T_1 storing value $\sigma_s(E_0)$ for address E_0/E_{\max} .

3.3.2. γ -photons

The sampling of the scattering direction for γ -photons requires to sample according to phase function P_{KN} .

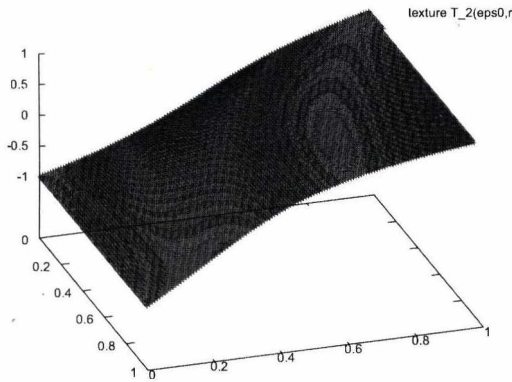


Figure 4: Texture T_2 storing $\cos\theta$ that is the solution of equation $v = CDF(\cos\theta, E_0)$ at texture coordinate pair $(E_0/E_{\max}, v)$.

The cumulative probability distribution with respect to $\cos\theta$ is

$$\begin{aligned}
 CDF(c, E_0) &= P(\cos\theta < c | \text{scattering}) = \\
 &= \int_0^c \int_{-1}^1 P_{KN}(\cos\theta, E_0) d\cos\theta d\phi = \\
 &= \frac{2\pi}{\sigma_s(E_0)} \int_{-1}^c \epsilon + \epsilon^3 - \epsilon^2(1 - \cos^2\theta) d\cos\theta.
 \end{aligned}$$

The cumulative distribution is just a one-variate integral for a given incident energy E_0 . Let us compute these integrals for regularly placed discrete samples of $E_0 \in [0, E_{\max}]$ and let us store the solution of equation

$$v = CDF(c, E_0)$$

in a 2D array, called texture $T(u, v)$, addressed by texture coordinates $u = E_0/E_{\max}$ and v . Note that this texture is independent of the material properties and should be computed only once during pre-processing.

During particle tracing the direction sampling is executed in the following way. Random or quasi-random sample $r_2 \in [0, 1)$ is obtained and we look up texture $T(u, v)$ with it and with the incident energy $u = E_0/E_{\max}$ and $v = r_2$ resulting in the cosine of the scattering angle. Note that the texture lookup automatically involves bi-linear interpolation of the pre-computed data at no additional cost. Using the Compton formula, the scattered energy is computed. The other spherical coordinate ϕ is sampled from uniform distribution, i.e. $\phi = 2\pi r_3$ where r_3 is a uniformly distributed random value in the unit interval. Let us establish a Cartesian coordinate system $\mathbf{i}, \mathbf{j}, \mathbf{k}$ where $\mathbf{k} = \vec{\omega}'$ is the incident direction, and:

$$\mathbf{i} = \frac{\mathbf{k} \times \mathbf{v}}{|\mathbf{k} \times \mathbf{v}|}, \quad \mathbf{j} = \mathbf{i} \times \mathbf{k}. \quad (7)$$

Here \mathbf{v} is an arbitrary vector that is not parallel with $\vec{\omega}'$. Using these unit vectors, the scattering direction $\vec{\omega}$ is:

$$\vec{\omega} = \sin\theta \cos\phi \mathbf{i} + \sin\theta \sin\phi \mathbf{j} + \cos\theta \mathbf{k}. \quad (8)$$

3.4. Gathering

Having obtained the voxel radiances the image is computed by accumulating the radiance values along each viewing ray.

4. Results

The proposed methods have been implemented in CUDA and their performance has been measured on an NVIDIA GeForce 9600 GPU at 600×600 resolution. The electron density of the head model is stored in a 128^3 resolution voxel array.

We followed photons until at most 5 scattering points. The results are shown by Figure 5 for light photons and by Figure 6 for γ -photons. Our implementation of the inversion method can trace 1 million photon paths of length at most 5 in about 12 sec on the frequency range of visible light and in 24 sec on the frequency range of γ -photons. The slow down of the simulation of γ -photons is due to the more complicated sampling of the photon energy and scattering direction. Interestingly, Woodcock tracking is not faster than the inversion method, although it reads the voxel volume less often. The reason of its poor behavior is that it has incoherent texture access patterns, which degrades texture cache utilization.

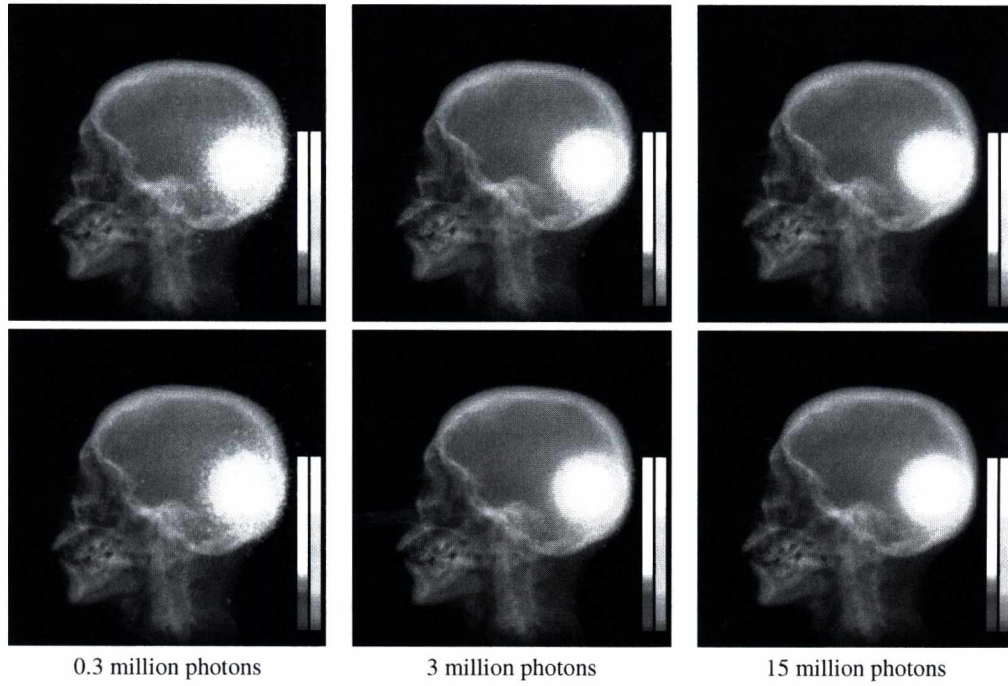


Figure 5: Light photon tracing with the inversion method (upper row) and with Woodcock tracking (lower row).

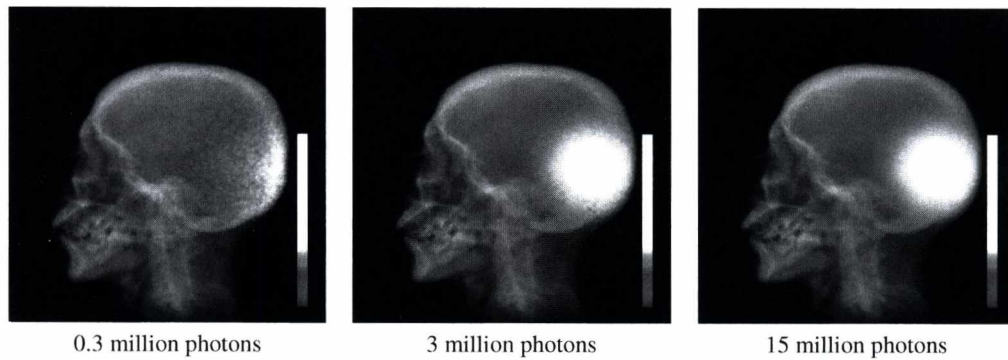


Figure 6: Light photon tracing with the inversion method.

5. Conclusions

This paper proposed an effective method to solve the radiative transport equation in inhomogeneous participating media on the GPU. We used Monte Carlo particle tracing, and implemented the algorithm on the massively parallel GPU architecture. This way, the transport simulation requires just a few seconds, i.e. it is almost interactive, which is a great speed up with respect to CPU simulations running for minutes and even for hours.

Acknowledgement

This work has been supported by the Teratomo project of the National Office for Research and Technology, OTKA K-719922 (Hungary), and by the Hungarian-Croatian Fund.

References

1. J. F. Blinn. Light reflection functions for simulation of clouds and dusty surfaces. In *SIGGRAPH '82 Proceedings*, pages 21–29, 1982.
2. E. Cerezo, F. Pérez, X. Pueyo, F. J. Seron, and F. X. Sillion. A survey on participating media rendering techniques. *The Visual Computer*, 21(5):303–328, 2005.
3. Balázs Csébfalvi. Prefiltered gaussian reconstruction for high-quality rendering of volumetric data sampled on a body-centered cubic grid. In *VIS '05: Visualization, 2005*, pages 311–318. IEEE Computer Society, 2005.
4. R. Geist, K. Rasche, J. Westall, and R. Schalkoff. Lattice-boltzmann lighting. In *Eurographics Symposium on Rendering, 2004*.
5. M. Harris and A. Lastra. Real-time cloud rendering. *Computer Graphics Forum*, 20(3):76–84, 2001.
6. H. W. Jensen and P. H. Christensen. Efficient simulation of light transport in scenes with participating media using photon maps. *SIGGRAPH '98 Proceedings*, pages 311–320, 1998.
7. H. W. Jensen, S. Marschner, M. Levoy, and P. Hanrahan. A practical model for subsurface light transport. *Computer Graphics (SIGGRAPH 2001 Proceedings)*, 2001.
8. J.T. Kajiya and B. Von Herzen. Ray tracing volume densities. In *Computer Graphics (SIGGRAPH '84 Proceedings)*, pages 165–174, 1984.
9. Joe Kniss, Simon Premoze, Charles Hansen, and David Ebert. Interactive translucent volume rendering and procedural modeling. In *VIS '02: Proceedings of the conference on Visualization '02*, pages 109–116, Washington, DC, USA, 2002. IEEE Computer Society.
10. Srinivasa G. Narasimhan and Shree K. Nayar. Shedding light on the weather. In *CVPR 03*, pages 665–672, 2003.
11. Vincent Pegoraro and Steven G. Parker. An analytical solution to single scattering in homogeneous participating media. *Computer Graphics Forum*, 28, 2009.
12. Feng Qiu, Fang Xu, Zhe Fan, and Neophytou Neophytos. Lattice-based volumetric global illumination. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1576–1583, 2007. Fellow-Arie Kaufman and Senior Member-Klaus Mueller.
13. H. E. Rushmeier and K. E. Torrance. The zonal method for calculating light intensities in the presence of a participating medium. In *SIGGRAPH 87*, pages 293–302, 1987.
14. Jos Stam. Multiple scattering as a diffusion process. In *Eurographics Rendering Workshop*, pages 41–50, 1995.
15. Bo Sun, Ravi Ramamoorthi, Srinivasa G. Narasimhan, and Shree K. Nayar. A practical analytic single scattering model for real time rendering. *ACM Trans. Graph.*, 24(3):1040–1049, 2005.
16. L. Szirmay-Kalos. *Monte-Carlo Methods in Global Illumination — Photo-realistic Rendering with Randomization*. VDM, Verlag Dr. Müller, Saarbrücken, 2008.
17. L. Szirmay-Kalos, G. Liktó, T. Umenhoffer, B. Tóth, S. Kumar, and G. Lupton. Parallel solution to the radiative transport. In Weiskopf Comba, Debattista, editor, *Eurographics Symposium on Parallel Graphics and Visualization*, pages 95–102, 2009.
18. L. Szirmay-Kalos, M. Sbert, and T. Umenhoffer. Real-time multiple scattering in participating media with illumination networks. In *Eurographics Symposium on Rendering*, pages 277–282, 2005.
19. L. Szirmay-Kalos, L. Szécsi, and M. Sbert. *GPU-Based Techniques for Global Illumination Effects*. Morgan and Claypool Publishers, San Rafael, USA, 2008.
20. E. Woodcock, T. Murphy, P. Hemmings, and S. Longworth. Techniques used in the GEM code for Monte Carlo neutronics calculation. In *Proc. Conf. Applications of Computing Methods to Reactors, ANL-7050*, 1965.
21. C. N. Yang. The Klein-Nishina formula & quantum electrodynamics. *Lect. Notes Phys.*, 746:393–397, 2008.
22. Kun Zhou, Zhong Ren, Stephen Lin, Hujun Bao, Baining Guo, and Heung-Yeung Shum. Real-time smoke rendering using compensated ray marching. *ACM Trans. Graph.*, 27(3):36, 2008.

Recent Results on Optimal Regular Volume Sampling

Balázs Cs. Bfalvi¹, Balázs Domonkos², Viktor Vad¹

¹ Budapest University of Technology and Economics

² Mediso Medical Imaging Systems, Hungary

Abstract

In this paper, we overview our recent results in the area of optimal regular volume sampling. First, we show that our prefiltered B-spline reconstruction technique can be efficiently used for quasi-interpolation on the optimal Body-Centered Cubic (BCC) lattice. This approach provides higher visual quality than the non-separable box-spline filtering previously proposed for BCC-sampled data. Afterwards, we demonstrate that a volumetric data set acquired on a traditional Cartesian Cubic (CC) lattice is worthwhile to upsample on a higher-resolution BCC lattice in the frequency domain. The obtained BCC-sampled volume representation is then interactively rendered by using a GPU-accelerated trilinear B-spline reconstruction. Although this technique doubles the storage requirements, it ensures similar reconstruction quality as a cubic filtering on the original CC-sampled data, but for an order of magnitude lower computational cost. Last but not least, we show that the Filtered Back-Projection (FBP) algorithm can be adapted to acquire volumetric data directly on a BCC lattice.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism; I.4.5 [Image Processing and Computer Vision]: Reconstruction; I.4.10 [Image Processing and Computer Vision]: Volumetric Image Representation

1. Introduction

In the last couple of years, the application of BCC sampling in volume modeling and visualization received increased attention. It has been shown that the theoretical advantages of BCC sampling can be exploited also in practice by using appropriate reconstruction schemes^{11, 4, 10, 5, 9, 13}. Unlike the CC lattice, the BCC lattice is optimal for sampling spherically band-limited signals^{23, 19}. Therefore, to perfectly reconstruct such a signal from its discrete representation, about 30% fewer samples per unit volume have to be taken on a BCC lattice than on an equivalent CC lattice. Although practical signals can hardly be considered band-limited, it is still favorable if the sampling scheme does not prefer drastically any specific direction. For example, images generated from CC-sampled volumetric data usually show axis-aligned staircase aliasing even if a high-quality filter is applied for the reconstruction⁶. Despite this drawback, the CC lattice is still a de facto standard in 3D data acquisition and visualization mainly because of efficiency reasons. The CC samples can be easily stored and referenced in a 3D array, and the resampling can be efficiently implemented by using a fast separable filtering. In order to suppress the axis-aligned arti-

facts, recently a non-separable filter has been proposed even for reconstruction on the separable CC lattice¹². Nevertheless, the evaluation of this seven-directional box-spline filter is not supported by the current GPUs and also its CPU implementation is rather inefficient compared to that of the popular tricubic B-spline resampling. As an alternative solution, we proposed to sample the original signal on an optimal BCC lattice and to apply tensor-product B-spline filters for the continuous reconstruction¹⁰. The B-spline filtering on the BCC lattice can be efficiently implemented on a current GPU and ensures an isotropic suppression of the postaliasing effects. Recently, we investigated the practical utility of this approach from several aspects^{7, 8}. The results of this investigation are summarized in this paper.

2. Related Work

Based on the arguments of Petersen and Middleton¹⁹, Theußl et al.²³ were first to recommend BCC sampling for volume modeling and visualization. One of the most important aspects of rendering BCC-sampled data is the continuous reconstruction from the discrete samples. For the CC lattice, reconstruction filters are usually designed in 1D

and extended to 3D by a separable tensor-product extension¹⁷. However, the advantageous properties of a 1D filter are not necessarily inherited in 3D if its separable extension is used for reconstruction on the non-separable BCC lattice. As an alternative, a spherical extension¹⁵ might seem to be a natural choice especially for the BCC lattice, which is optimal for sampling spherically band-limited signals. Nevertheless, the main disadvantage of this approach is that in the frequency response of a spherically extended filter, zero-crossings at the points of the dual Face-Centered Cubic (FCC) lattice are difficult to guarantee¹¹. These zero-crossings are necessary for a reconstruction of high approximation order that can perfectly reproduce polynomials up to a certain degree^{21, 2, 3}. Although early reconstruction methods still attempted to use either separable or spherically extended filters on the BCC lattice, the results did not reflect the theoretical benefits of optimal sampling. For example, the application of spherical filters²³ resulted in rather blurry images, while the separable sheared trilinear interpolation^{22, 16} led to an anisotropic reconstruction. The first non-separable filters that are tailored to the geometry of the BCC lattice were proposed by Entezari et al.¹¹. In a CPU implementation¹³, their linear and quintic box-spline filters were twice as fast to evaluate as the equivalent B-spline filters on the CC lattice, which are the trilinear and tricubic B-splines respectively. However, in a GPU-accelerated ray-casting application¹⁴, the non-separable box-spline filtering on the BCC lattice was still slower than the separable B-spline filtering on the CC lattice²⁰. Another family of non-separable filters is represented by the BCC-splines⁵, which generalize the Hex-splines for the BCC lattice. The Hex-splines were originally proposed by Van De Ville et al.²⁵ for the hexagonal lattice, which is optimal for sampling circularly band-limited 2D signals. The BCC-splines were evaluated by a discrete approximation and their analytical formulas are not known yet. Csébfalvi recommended a prefiltered Gaussian reconstruction scheme⁴ adapting the principle of *generalized interpolation*¹ to the BCC lattice. This method was extended also to the B-spline family of filters¹⁰, and a fast GPU implementation was proposed. According to our best knowledge, the prefiltered B-spline reconstruction is the fastest technique up to now that can be used for rendering BCC-sampled data interactively. In this paper we present two improvements. First, we demonstrate how to guarantee certain polynomial approximation orders by using appropriate discrete prefilters. Second, we show that the prefiltered B-spline reconstruction is worthwhile to apply on a BCC-sampled volume that is obtained by upsampling an originally CC-sampled data. Additionally, we adapt the FBP algorithm to naturally acquire data on a BCC lattice.

3. Prefiltered B-Spline Reconstruction

The Prefiltered B-Spline Reconstruction (PBSR) scheme¹⁰ exploits that the BCC lattice consists of two interleaved CC lattices, where the second CC lattice is translated by a vector

$[\frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$ (see Fig. 1). Thus, the reconstruction can be performed separately for these two CC lattices by using the standard trilinear or tricubic B-spline resampling twice and averaging their contributions. This evaluation is equivalent to the convolution of the BCC samples with the given B-spline filter kernel. The trilinear and tricubic filters are defined as the tensor-product extensions of the B-splines of orders one and three, therefore we denote them by β^1 and β^3 respectively. The tricubic B-spline can also be obtained by convolving the trilinear B-spline with itself: $\beta^3 = \beta^1 * \beta^1$ ²⁴.

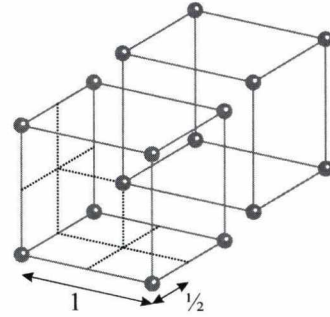


Figure 1: The BCC lattice as two interleaved CC lattices. The red CC lattice is translated by a vector $[\frac{1}{2}, \frac{1}{2}, \frac{1}{2}]$.

In order to better fit the separable B-spline resampling to the geometry of the BCC lattice, a non-separable prefiltering has been proposed¹⁰. Unfortunately, the scheme of *generalized interpolation*¹ cannot be applied, since a discrete prefiltering that makes the B-spline reconstruction interpolating on the BCC lattice theoretically does not exist. This claim can be justified by a frequency-domain analysis. The Fourier transform of the trilinear B-spline β^1 is

$$\hat{\beta}^1(\omega) = [\text{sinc}(\omega_x)\text{sinc}(\omega_y)\text{sinc}(\omega_z)]^2, \quad (1)$$

where $\omega = [\omega_x, \omega_y, \omega_z]^T$ and $\text{sinc}(t) = \frac{\sin(t/2)}{t/2}$. For *generalized interpolation*, the prefilter should be the inverse of the BCC-sampled reconstruction kernel. If the trilinear kernel (which is not interpolating on the BCC lattice) is sampled on the BCC lattice, its Fourier transform $\hat{\beta}^1(\omega)$ is replicated around the points of the dual FCC lattice:

$$\beta^1(\mathbf{x}) \sum_{\mathbf{k} \in \mathbb{Z}^3} \delta(\mathbf{x} - \mathbf{G}\mathbf{k}) \Leftrightarrow |\det \hat{\mathbf{G}}| \sum_{\mathbf{k} \in \mathbb{Z}^3} \hat{\beta}^1(\omega - 2\pi\hat{\mathbf{G}}\mathbf{k}), \quad (2)$$

where the generator matrix of the BCC lattice¹³ is

$$\mathbf{G} = \frac{1}{2} \begin{bmatrix} 1 & -1 & -1 \\ -1 & 1 & -1 \\ -1 & -1 & 1 \end{bmatrix},$$

and the generator matrix of the dual FCC lattice is

$$\hat{\mathbf{G}} = \mathbf{G}^{-T} = \begin{bmatrix} 0 & -1 & -1 \\ -1 & 0 & -1 \\ -1 & -1 & 0 \end{bmatrix}.$$

Since (2) equals zero for any $\omega = 2\pi[u, v, w]^T$, where $u, v, w \in \mathbb{Z}$ and $u + v + w$ is odd, the BCC-sampled trilinear kernel is not invertible. Similarly, the BCC-sampled tricubic B-spline is not invertible either as its Fourier transform is

$$|\det \hat{\mathbf{G}}| \sum_{\mathbf{k} \in \mathbb{Z}^3} \hat{\beta}^3(\omega - 2\pi \hat{\mathbf{G}}\mathbf{k}), \quad (3)$$

where $\hat{\beta}^3(\omega) = \hat{\beta}^1(\omega) \cdot \hat{\beta}^1(\omega)$. In the previous work¹⁰, a frequency-domain solution has been proposed that mimics the prefiltering for generalized interpolation such that the approximation error at the lattice points, which is controlled by a free parameter, can be made arbitrarily small. Nevertheless, this approach does not ensure a certain order of accuracy at an arbitrary sample position.

4. Quasi-Interpolation on the BCC Lattice

An originally continuous 3D function $f(\mathbf{x})$ can be reconstructed from its discrete samples by a linear combination of shifted copies of a basis function $\varphi(\mathbf{x})$:

$$f(\mathbf{x}) \approx \tilde{f}(\mathbf{x}) = \sum_{\mathbf{k} \in \mathbb{Z}^3} c[\mathbf{k}] \varphi(\mathbf{x} - \mathbf{G}\mathbf{k}), \quad (4)$$

where \mathbf{G} is the generator matrix of the sampling lattice and the coefficients $c[\mathbf{k}]$ are determined from the discrete samples $f[\mathbf{k}] = f(\mathbf{G}\mathbf{k})$. If $\tilde{f}(\mathbf{G}\mathbf{k}) = f[\mathbf{k}]$ for each $\mathbf{k} \in \mathbb{Z}^3$, the reconstruction is an interpolation, that is, the accurate function values are reproduced at the lattice points. The interpolation condition is trivially satisfied if the reconstruction kernel φ is interpolating on the given lattice ($\varphi(\mathbf{G}\mathbf{k}) = \delta_{\mathbf{k}}$) and $c[\mathbf{k}] = f[\mathbf{k}]$. If φ is not interpolating then generalized interpolation¹ can be applied to calculate coefficients $c[\mathbf{k}]$ such that the interpolation condition is still satisfied.

In volume-rendering applications, interpolation is not necessarily the best way of reconstruction. The discrete volume representation has to be resampled at arbitrary positions, therefore it is favorable to guarantee a uniform distribution of the approximation error in order to avoid visual artifacts. Forcing a higher-degree approximate function to pass exactly through the sample values, however, usually leads to overshooting that appears as a ringing artifact in the images generated by volume rendering¹⁵.

Another alternative for function reconstruction is quasi-interpolation^{2,3}, which sacrifices the interpolation condition to increase the global approximation quality such that $\tilde{f}(\mathbf{x})$ well approximates $f(\mathbf{x})$ at an arbitrary position \mathbf{x} . An excellent framework on quasi-interpolation on the 2D hexagonal lattice was published by Condat and Van De Ville³. It is important to note that their conditions for quasi-interpolation are not restricted to 2D lattices but applicable to any multi-dimensional lattice. By definition, quasi-interpolation of order L perfectly reconstructs a polynomial of degree at most $L - 1$. This is usually achieved by a discrete prefiltering, where coefficients $c[\mathbf{k}]$ are produced by convolving the data values $f[\mathbf{k}]$ with a discrete prefilter p . Quasi-interpolation

of order L is equivalent to the following condition³ for the prefilter p :

$$\hat{p}(\omega) \hat{\varphi}(\omega - 2\pi \hat{\mathbf{G}}\mathbf{k}) = \delta_{\mathbf{k}} + O(\|\omega\|^L) \text{ for every } \mathbf{k} \in \mathbb{Z}^3, \quad (5)$$

where \hat{p} and $\hat{\varphi}$ are the Fourier transforms of the prefilter p and the reconstruction filter φ respectively. Matrix $\hat{\mathbf{G}}$ is the generator matrix corresponding to the dual (the Fourier transform) of the sampling lattice. Additionally, it is necessary for a quasi-interpolation of order L that the reconstruction kernel φ satisfies the well-known Strang-Fix conditions²¹:

$$\hat{\varphi}(\mathbf{0}) \neq 0 \text{ and } \hat{\varphi}(\omega - 2\pi \hat{\mathbf{G}}\mathbf{k}) = O(\|\omega\|^L) \text{ for every } \mathbf{k} \neq \mathbf{0}. \quad (6)$$

Thus, the frequency response of the reconstruction filter has to guarantee zero-crossings of order at least L at the dual lattice points except the origin. The order of such a zero-crossing is referred to as the number of *vanishing moments*^{11,13}, and it is not necessarily the same for all the dual lattice points. The *approximation power* L of a filter is defined as the minimal number of its vanishing moments.

4.1. Quasi-Interpolation versus Generalized Interpolation

Unlike the tensor-product B-splines, the non-separable box splines can be used also for interpolation on the BCC lattice. The linear box spline is interpolating itself and the quintic box spline can be made interpolating by an appropriate discrete prefiltering⁹. This is definitely an important theoretical advantage of the non-separable box splines. However, we recently derived a prefilter of IIR for quasi-interpolating Prefiltered Cubic B-Spline Reconstruction (PCBSR)⁸, which makes the pass-band behavior similar to that of the interpolating Prefiltered Quintic Box-spline Reconstruction (PQBXR), but in the stop band, well preserves the isotropic antialiasing effect of the tricubic B-spline.

Applying the scheme of generalized interpolation¹ on the BCC lattice, the data samples have to be convolved with the inverse of the BCC-sampled reconstruction kernel. Consequently, the Fourier transform of the discrete prefilter q that makes the quintic box-spline reconstruction interpolating is defined as follows:

$$\hat{q}(\omega) = \frac{1}{\hat{s}(\omega)} = \frac{1}{\sum_{\mathbf{k} \in \mathbb{Z}^3} \hat{M}_{\Xi^2}(\omega - 2\pi \hat{\mathbf{G}}\mathbf{k})}, \quad (7)$$

where $\hat{s}(\omega)$ is the Fourier transform of the discrete deconvolution filter $s[\mathbf{k}]$ produced by sampling the quintic box-spline M_{Ξ^2} on the BCC lattice and normalizing the BCC samples: $s[\mathbf{k}] = |\det \mathbf{G}| M_{\Xi^2}(\mathbf{G}\mathbf{k})$. In Equation (7), the Fourier transform of the quintic box spline is¹³:

$$\hat{M}_{\Xi^2}(\omega) = \hat{M}_{\Xi}(\omega)^2 = \prod_{k=1}^4 \text{sinc}^2(\xi_k^T \omega), \quad (8)$$

where vectors ξ_k are defined by the directions of the first

nearest BCC neighbors:

$$\Xi = [\xi_1, \xi_2, \xi_3, \xi_4] = \frac{1}{2} \begin{bmatrix} 1 & -1 & -1 & 1 \\ -1 & 1 & -1 & 1 \\ -1 & -1 & 1 & 1 \end{bmatrix}.$$

Note that the approximation power of the tricubic B-spline on the BCC lattice is $L = 4$ (see the details in Section 4.2), therefore it can potentially be applied for quasi-interpolation of order four by using an appropriate discrete prefiltering. The key idea for finding a good IIR prefilter also for PCBSR was to utilize the advantageous properties of prefilter q . More concretely, we proposed to perform prefiltering by q to fully exploit the approximation power of the tricubic B-spline. Our approach is based on the following theorem.

Theorem 4.1 By convolving q with itself, a prefilter is obtained that makes PCBSR quasi-interpolating of order four on the BCC lattice.

Proof It is well-known that generalized interpolation, which combines a discrete IIR prefilter p with a reconstruction filter ϕ of approximation power L , satisfies the following satisfactory condition for quasi-interpolation of order L ^{2,3}:

$$\frac{1}{\hat{p}(\omega)} = \hat{\phi}(\omega) + O(\|\omega\|^L). \quad (9)$$

Since the approximation power of the quintic box spline is $L = 4$ ¹³, using q as prefilter p and M_{Ξ^2} as reconstruction filter ϕ , we obtain

$$\frac{1}{\hat{q}(\omega)} = \hat{s}(\omega) = \hat{M}_{\Xi^2}(\omega) + O(\|\omega\|^4). \quad (10)$$

Therefore, the Taylor series expansion of M_{Ξ^2} around $\omega = 0$ gives

$$\frac{1}{\hat{q}(\omega)} = 1 - \frac{1}{12}(\omega_x^2 + \omega_y^2 + \omega_z^2) + O(\|\omega\|^4). \quad (11)$$

The direct use of q would not make PCBSR quasi-interpolating of order four. However, using $q * q$ as prefilter p and β^3 as reconstruction filter ϕ , condition (9) for $L = 4$ is already satisfied:

$$\begin{aligned} \frac{1}{\hat{q}(\omega)^2} &= [1 - \frac{1}{12}(\omega_x^2 + \omega_y^2 + \omega_z^2) + O(\|\omega\|^4)]^2 \\ &= 1 - \frac{1}{6}(\omega_x^2 + \omega_y^2 + \omega_z^2) + O(\|\omega\|^4) = \hat{\beta}^3(\omega) + O(\|\omega\|^4). \end{aligned} \quad (12)$$

□

4.2. Frequency-Domain Analysis

It is easy to see that the approximation powers of the trilinear and tricubic B-splines are the same as that of the linear and quintic box splines, which are two and four respectively. Their energy in the stop band, however, is distributed rather differently. If the frequency-domain decay of a reconstruction filter is short, like in case of B-splines and box splines,

then mainly those aliasing spectra contribute to the postaliasing effect that are the closest to the primary spectrum. These aliasing spectra are located around the nearest twelve FCC lattice points, which are $[\pm 2\pi, \pm 2\pi, 0]$, $[\pm 2\pi, 0, \pm 2\pi]$, and $[0, \pm 2\pi, \pm 2\pi]$. At these aliasing frequencies, the trilinear and tricubic B-splines ensure *four* and *eight* vanishing moments respectively. In contrast, at these points of the frequency domain, the linear and quintic box splines guarantee just the minimal number of their vanishing moments, which are *two* and *four* respectively. As a consequence, based on the theory developed by Entezari et al.¹², the tensor-product B-splines can suppress the postaliasing effect more efficiently on the BCC lattice than the non-separable box splines of the same approximation powers.

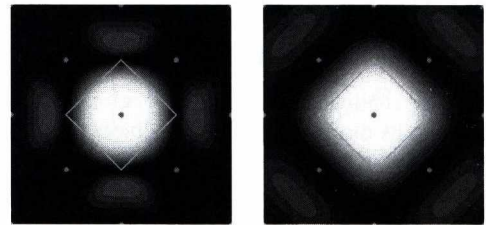


Figure 2: Resultant frequency responses after an IIR prefiltering. Left: PCBSR combined with our IIR prefilter $q * q$. Right: Interpolating PQBXR. The images represent the central slice ($\omega_z = 0$) of the frequency response in the domain $[-4\pi, 4\pi]^2$, where the green rhombus depicts the border of the pass band and the red dots depict the FCC lattice points.

Fig. 2 shows the central slice of the resultant frequency response for both the quasi-interpolating PCBSR and the interpolating PQBXR. In order to emphasize the stop-band behavior, we displayed the frequency responses with gray levels proportional to the cube root of the original values. PCBSR ensures much less postaliasing especially around the nearest FCC lattice points, whereas PQBXR provides a little bit better pass-band behavior with less oversmoothing.

4.3. Experimental Analysis

We have compared our PCBSR to PQBXR by rendering BCC-sampled representations of the well-known Marschner-Lobb signal¹⁵. In order to implement the deconvolution, in both cases we used a frequency-domain solution proposed by Csébfalvi and Domonkos⁹. The results are shown in Fig. 3. Note that, using our quasi-interpolating PCBSR, the circular level lines are almost perfectly reconstructed even from the lower-resolution volume representation that samples the test signal near the Nyquist limit. In contrast, the interpolating PQBXR produces visible direction-dependent aliasing unless the signal is sampled at a higher resolution. Although the average angular error of the gradients calculated by PCBSR is slightly higher (see Fig. 4), the error is much more uniformly distributed than in case of PQBXR.

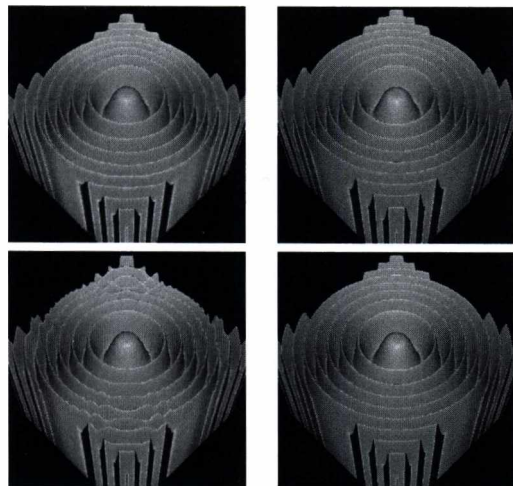


Figure 3: Reconstruction of the Marschner-Lobb signal using PCBSR with our IIR prefilter $q * q$ (upper row) and the interpolating PQBXR (lower row). Left column: Reconstruction from $32 \times 32 \times 32 \times 2$ BCC samples. Right column: Reconstruction from $64 \times 64 \times 64 \times 2$ BCC samples.

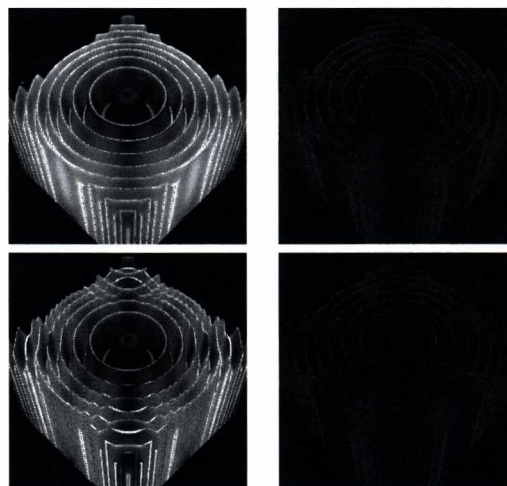


Figure 4: Angular error of the gradients reconstructed by PCBSR combined with our IIR prefilter $q * q$ (upper row) and the interpolating PQBXR (lower row). Angular error of 30 degrees is mapped to white, whereas angular error of zero degree is mapped to black. Left column: Reconstruction from $32 \times 32 \times 32 \times 2$ BCC samples. Right column: Reconstruction from $64 \times 64 \times 64 \times 2$ BCC samples.

5. Frequency-Domain Upsampling on a BCC Lattice

Our quasi-interpolating PCBSR technique has been proposed for volumetric data acquired on an optimal BCC lattice⁸. However, the idea of BCC sampling for 3D data acquisition is relatively new, therefore the recent 3D scanning technologies, such as CT or MRI, usually produce data on a traditional CC lattice. In this section, we demonstrate that it is still worthwhile to convert CC-sampled data onto a BCC representation⁷.

For real-time visualization of CC-sampled volume data, usually the trilinear filter is used, since its evaluation is directly supported by the recent GPUs. Nevertheless, the trilinear filter provides poor quality if the data is not sampled at an appropriate frequency. In order to increase the reconstruction quality, two fundamentally different strategies can be followed. The first strategy is to apply higher-order filters¹⁷, such as the cubic B-spline or the Catmull-Rom spline. These filters, however, are much more expensive computationally than the popular trilinear filter. A trilinear sample is calculated from the eight nearest voxels, whereas the cubic filters take the nearest 64 voxels into account. In practice, a filter of a larger support can drastically decrease the rendering performance because of the costly cache misses. Although it has been shown that a cubic filtering can be efficiently implemented on the recent GPUs by calculating a weighted average of eight trilinear texture samples²⁰, the trilinear filtering is still significantly faster.

The second strategy for improving the quality of local resampling is to combine the complementary advantages of spatial-domain and frequency-domain techniques. In a

preprocessing step, the initial volume is upsampled in the frequency-domain implementing the *sinc* interpolation by either zero padding or phase shifts¹⁸. After the preprocessing, the obtained higher-resolution volume is resampled in the spatial domain by using a filter of a smaller support. Generally, the smaller the support of a filter, the higher is its postaliasing effect¹⁵. However, the upsampling can potentially compensate the higher postaliasing effect of a cheap reconstruction filter. The major drawback of this approach is that the memory requirements are drastically increased. For example, if the volume resolution is doubled along each axis, the number of voxels is increased by a factor of eight, which might be prohibitive in practical applications.

In order to attain the highest reconstruction quality for a fixed storage overhead, we recently proposed a frequency-domain upsampling of CC-sampled data on an optimal BCC lattice rather than on a higher-resolution CC lattice⁷. The obtained BCC representation is rendered by using a simple GPU-accelerated trilinear B-spline reconstruction. Although this approach doubles the storage requirements, it provides similar quality as the most popular cubic filters, but for a significantly lower computational overhead.

5.1. Preprocessing

As it is demonstrated in Figure 1, the BCC lattice consists of two interleaved CC lattices. Assume that the volume is originally sampled on the blue CC lattice. If this lattice sufficiently samples the underlying signal, the samples on the red

lattice can be obtained by a *sinc* interpolation implemented as a frequency domain phase shift. Based on the shifting theorem, the DFT $F_{u,v,w}^{(0)}$ of the blue samples $f_{i,j,k}^{(0)}$ is modulated by $e^{-i\pi(u+v+w)}$. The red samples $f_{i,j,k}^{(1)}$ are obtained by transforming the result back into the spatial domain. In this way, a BCC-sampled volume data is produced, which oversamples the original signal. Although the BCC upsampling yields a redundant volume representation, it is still more reasonable than an upsampling on an equivalent CC lattice, which would result in 30% more samples.

5.2. Rendering

The upsampling makes sense if the obtained BCC-sampled volume can be resampled by a computationally cheap filter more efficiently than the original CC-sampled volume using a higher-order filter. Additionally, the reconstruction quality is required to be at least competitive. Therefore, the choice of the continuous reconstruction filter is of crucial importance in our method.

In a CPU implementation, the quintic box-spline filtering on a BCC lattice is twice as fast as an equivalent tricubic B-spline filtering on a CC lattice¹³. Nevertheless, in a GPU implementation, the tricubic B-spline filter is still faster to evaluate, since unlike the quintic box-spline filter, it can utilize the direct hardware support for trilinear texture fetching¹⁴. As we focused on GPU-based real-time volume rendering, we rejected the quintic box spline as an alternative. A GPU-accelerated tricubic B-spline reconstruction is slower on a BCC lattice than on a CC lattice¹⁰, therefore this alternative was also rejected. In the state of the art, there were really just two linear filters left that could be considered for efficiently rendering BCC-sampled data. One of them is the four-directional linear box spline¹⁴, and the other one is the trilinear B-spline¹⁰. In order to analyze these filters in terms of both quality and efficiency, we implemented a texture-based isosurface rendering application using optimized shader codes for the resampling.

We rendered a measured data set, which is a CT scan of a carp. First we tried to reconstruct the skeleton from the initial $128 \times 128 \times 256$ CC samples. Figure 5 shows that a simple trilinear interpolation causes severe staircase aliasing. Using a tricubic B-spline reconstruction, the staircase artifacts are avoided, but the fine details are removed. In contrast, an interpolating prefiltered tricubic B-spline reconstruction gives an excellent result. Note that the linear reconstruction schemes combined with our BCC upsampling guarantee similar visual quality as a cubic filtering on the initial CC-sampled data. On the BCC lattice, the trilinear B-spline filter reduces the postaliasing effect more efficiently than the linear box-spline filter, but the latter is better in reconstructing the fine details. Concerning the efficiency, the trilinear B-spline filtering is an order of magnitude faster than the linear box-spline filtering or a tricubic B-spline filtering on the CC lattice.

6. Tomography Reconstruction on the BCC Lattice

In order to naturally acquire volumetric data on a BCC lattice, we adapted the well-known FBP algorithm and tested it on a real CT scanner provided by Mediso Medical Imaging Systems. We generated theoretically equivalent CC-sampled and BCC-sampled volume representations and rendered the obtained data sets by using the trilinear and tricubic B-spline filters. Figure 6 shows the reconstruction of a mouse skeleton from the discrete samples. Note that, due to its optimality, the BCC sampling requires about 30% fewer samples to store. Nevertheless, the quality of the images produced from the BCC-sampled data is still higher than that of the images produced from the CC-sampled data.

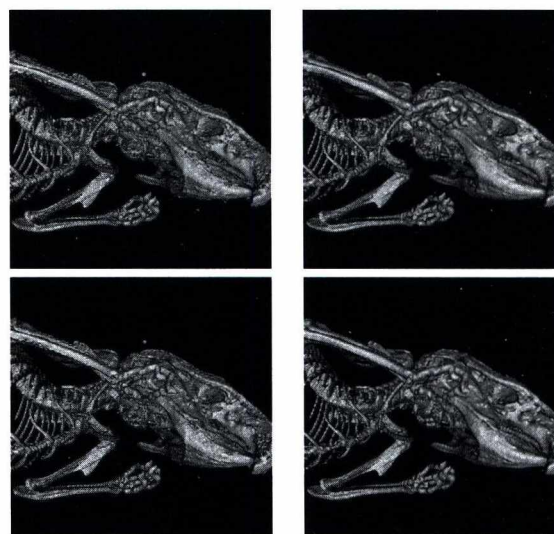


Figure 6: Trilinear (upper row) and tricubic (lower row) B-spline reconstructions of a mouse skeleton. Left column: Reconstruction from $424 \times 424 \times 509$ CC samples. Right column: Reconstruction from $300 \times 300 \times 360 \times 2$ BCC samples.

7. Conclusion

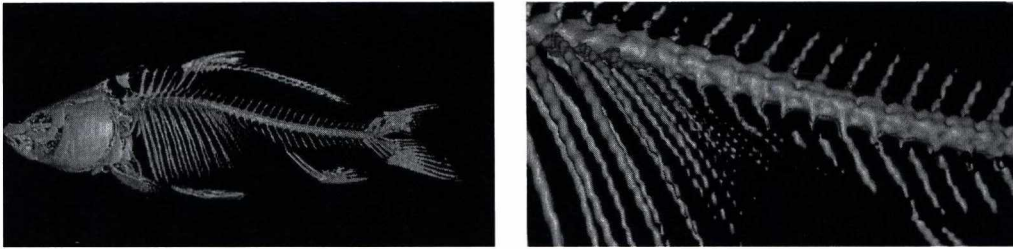
In this paper, we have summarized our recent results on optimal regular volume sampling. We demonstrated that the tensor-product B-spline filters represent a reasonable alternative of the non-separable box splines previously proposed for the BCC lattice from both theoretical and practical aspects. Theoretically, the trilinear and tricubic B-splines can provide the same polynomial approximation orders as the linear and quintic box splines respectively. From a practical point of view, unlike the non-separable box splines, the tensor-product B-splines can utilize the fast trilinear texture-fetching capability of the recent GPUs. Therefore, they can be used for interactively rendering BCC sampled data produced by either upsampling or tomography reconstruction.

Acknowledgements

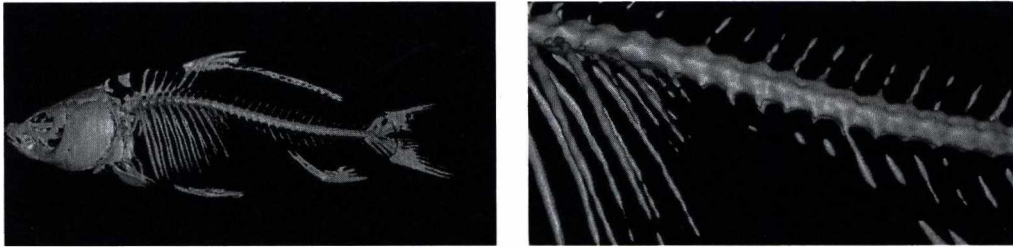
This work was supported by Mediso Medical Imaging Systems, the Hungarian National Office for Research and Technology (TECH 08/A2), the János Bolyai Research Scholarship of the Hungarian Academy of Sciences, and OTKA (F68945). The first author of this paper is a grantee of the János Bolyai Scholarship.

References

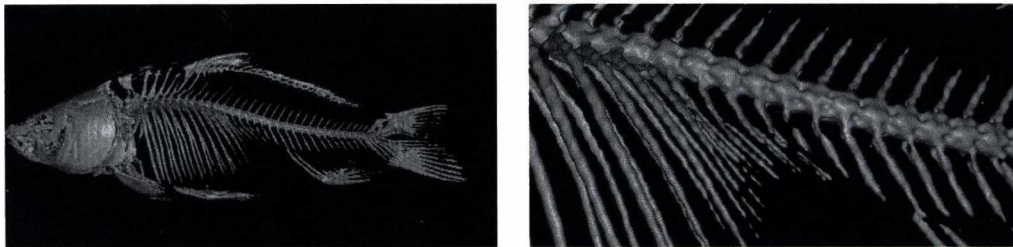
1. T. Blu, P. Th venaz, and M. Unser. Generalized interpolation: Higher quality at no additional cost. In *Proceedings of IEEE International Conference on Image Processing*, pages 667–671, 1999. 2, 3
2. L. Condat, T. Blu, and M. Unser. Beyond interpolation: Optimal reconstruction by quasi-interpolation. In *Proceedings of the IEEE International Conference on Image Processing*, pages 33–36, 2005. 2, 3, 4
3. L. Condat and D. Van De Ville. Quasi-interpolating spline models for hexagonally-sampled data. *IEEE Transactions on Image Processing*, 16(5):1195–1206, 2007. 2, 3, 4
4. B. Cs bfalvi. Prefiltered Gaussian reconstruction for high-quality rendering of volumetric data sampled on a body-centered cubic grid. In *Proceedings of IEEE Visualization*, pages 311–318, 2005. 1, 2
5. B. Cs bfalvi. BCC-splines: Generalization of B-splines for the body-centered cubic lattice. *Journal of WSCG*, 16(1–3):81–88, 2008. 1, 2
6. B. Cs bfalvi. An evaluation of prefiltered reconstruction schemes for volume rendering. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):289–301, 2008. 1
7. B. Cs bfalvi. Frequency-domain upsampling on a body-centered cubic lattice for efficient and high-quality volume rendering. In *Proceedings of Vision, Modeling, and Visualization*, 2009. 1, 5
8. B. Cs bfalvi. An evaluation of prefiltered B-spline reconstruction for quasi-interpolation on the body-centered cubic lattice. to appear in *IEEE Transactions on Visualization and Computer Graphics*, 2010. 1, 3, 5
9. B. Cs bfalvi and B. Domonkos. Pass-band optimal reconstruction on the body-centered cubic lattice. In *Proceedings of Vision, Modeling, and Visualization*, pages 71–80, 2008. 1, 3, 4
10. B. Cs bfalvi and M. Hadwiger. Prefiltered B-spline reconstruction for hardware-accelerated rendering of optimally sampled volumetric data. In *Proceedings of Vision, Modeling, and Visualization*, pages 325–332, 2006. 1, 2, 3, 6
11. A. Entezari, R. Dyer, and T. M Iler. Linear and cubic box splines for the body centered cubic lattice. In *Proceedings of IEEE Visualization*, pages 11–18, 2004. 1, 2, 3
12. A. Entezari and T. M Iler. Extensions of the Zwart-Powell box spline for volumetric data reconstruction on the Cartesian lattice. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of IEEE Visualization)*, 12(5):1337–1344, 2006. 1, 4
13. A. Entezari, D. Van De Ville, and T. M Iler. Practical box splines for reconstruction on the body centered cubic lattice. *IEEE Transactions on Visualization and Computer Graphics*, 14(2):313–328, 2008. 1, 2, 3, 4, 6
14. B. Finkbeiner, A. Entezari, T. M Iler, and D. Van De Ville. Efficient volume rendering on the body centered cubic lattice using box splines. *TR-CMPT2009-04, Graphics, Usability and Visualization (GrUVi) laboratory, Simon Fraser University*, 2009. 2, 6
15. S. Marschner and R. Lobb. An evaluation of reconstruction filters for volume rendering. In *Proceedings of IEEE Visualization*, pages 100–107, 1994. 2, 3, 4, 5
16. O. Mattausch. Practical reconstruction schemes and hardware-accelerated direct volume rendering on body-centered cubic grids. *Master's Thesis, Vienna University of Technology*, 2003. 2
17. T. M Iler, K. Mueller, Y. Kurzion, R. Machiraju, and R. Yagel. Design of accurate and smooth filters for function and derivative reconstruction. In *Proceedings of IEEE Symposium on Volume Visualization*, pages 143–151, 1998. 2, 5
18. A. V. Oppenheim and R. W. Schaffer. *Discrete-Time Signal Processing*. Prentice Hall Inc., Englewood Cliffs, 2nd edition, 1989. 5
19. D. P. Petersen and D. Middleton. Sampling and reconstruction of wave-number-limited functions in n-dimensional euclidean spaces. *Information and Control*, 5(4):279–323, 1962. 1
20. C. Sigg and M. Hadwiger. Fast third-order texture filtering. In *GPU Gems 2: Programming Techniques for High-Performance Graphics and General-Purpose Computation*, pages 313–329. Matt Pharr (ed.), Addison-Wesley, 2005. 2, 5
21. G. Strang and G. Fix. A Fourier analysis of the finite element variational method. In *Constructive Aspects of Functional Analysis*, pages 796–830, 1971. 2, 3
22. T. Theußl, O. Mattausch, T. M Iler, and M. E. Gr Iler. Reconstruction schemes for high quality raycasting of the body-centered cubic grid. *TR-186-2-02-11, Institute of Computer Graphics and Algorithms, Vienna University of Technology*, 2002. 2
23. T. Theußl, T. M Iler, and M. E. Gr Iler. Optimal regular volume sampling. In *Proceedings of IEEE Visualization*, pages 91–98, 2001. 1, 2
24. P. Th venaz, T. Blu, and M. Unser. Interpolation revisited. *IEEE Transactions on Medical Imaging*, 19(7):739–758, 2000. 2
25. D. Van De Ville, T. Blu, and M. Unser. Hex-splines: A novel spline family for hexagonal lattices. *IEEE Transactions on Image Processing*, 13(6):758–772, 2004. 2



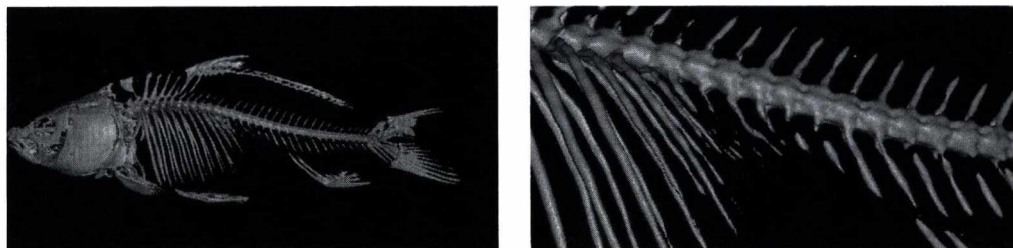
Trilinear B-spline reconstruction from $128 \times 128 \times 256$ CC samples.



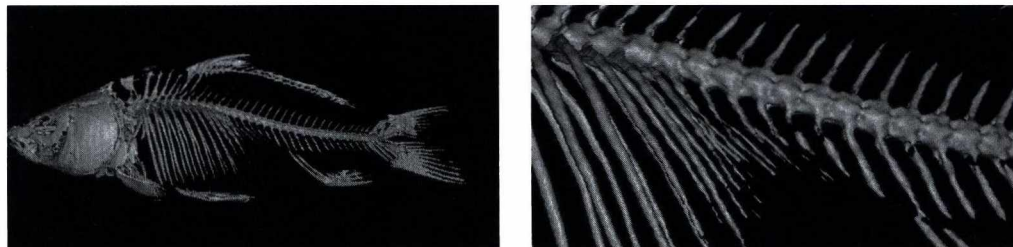
Tricubic B-spline reconstruction from $128 \times 128 \times 256$ CC samples.



Prefiltered tricubic B-spline reconstruction from $128 \times 128 \times 256$ CC samples.



Trilinear B-spline reconstruction from $128 \times 128 \times 256 \times 2$ BCC samples.



Linear box-spline reconstruction from $128 \times 128 \times 256 \times 2$ BCC samples.

Figure 5: Reconstructing the skeleton of a carp from $128 \times 128 \times 256$ CC samples and from $128 \times 128 \times 256 \times 2$ BCC samples obtained by a frequency-domain upsampling. The data set is courtesy of <http://www9.informatik.uni-erlangen.de/External/vollib>.

LDNI alapú söpört térfogat generálása anyageltávolítás jellegű gépészeti szimulációhoz

Tukora B.¹ Szalay T.²

¹ Műszaki Informatika Tanszék, Pollack Mihály Műszaki Kar, Pécsi Tudományegyetem, Pécs

² Gyártástudomány és -technológia Tanszék, Gépészmérnöki Kar, Budapesti Műszaki és Gazdaságtudományi Egyetem, Budapest

Abstract

A Pécsi Tudományegyetem Pollack Mihály Műszaki Karának Műszaki Informatika Tanszékén az elmúlt években kifejlesztésre került egy olyan gépészeti szimulációs eljárás, amely teljes egészében a grafikus hardveren fut, nagyfokú párhuzamosított végrehajtást téve ezzel lehetővé. Az eljárás specialitása, hogy bizonyos anyageltávolítás jellegű szimulációknál LDNI (Layered Depth-Normal Images) képeket használ a szerszám alakjának, vagy a szerszám által a szerszámpálya mentén végigsöpört térfogatnak a leírásához. Ez utóbbi generálásának hatékony módja kiemelt kutatási témát jelent a CAM rendszerek fejlesztőinek részére, újabb és újabb megoldások jelennek meg a szakirodalomban. Jelen írás a söpört térfogat generálásának grafikus hardver alapú módszerét mutatja be. E módszer speciálisan a korábban kifejlesztett gépészeti szimulációs eljáráshoz készült annak felgyorsítására; nem ad általános érvényű megoldást a problémára, de a söpört térfogat generálásának egy újszerű módját mutatja be, amely a számítógépes grafika hagyományos algoritmusaira épül.

1. Bevezetés

Az anyageltávolítás jellegű gépészeti szimulációk során olyan megmunkálásokat modellezünk, ahol egy adott formájú vágószerszám egy bizonyos szerszámpálya mentén haladva anyagot távolít el a munkadarabból, így alakítva ki annak végső formáját. Ilyen műveletek például a marás vagy esztergálás. Ezek a szimulációk a CAM (Computer Aided Manufacturing) rendszerek szerves részei.

Kutatásunk során az anyagleválasztás szimulációt GPGPU alapokon végezzük el. Ez azt jelenti, hogy a munkadarab alakjának reprezentációja, az anyageltávolítási számítások és a megjelenítés mind az általános célú számításokra alkalmas grafikus hardver (GPGPU) hatáskörébe tartoznak. A módszer előnye abban rejlik, hogy a nagyrészt szekvenciális programfuttatást végző CPU-tól teljes egészében átveszi a GPU a feladatokat, nagyfokú párhuzamos végrehajtást biztosítva, a CPU-GPU közötti adatmozgás kiküszöbölésével pedig megszűnik a korlátozott sávsebesség okozta szűk keresztmetszet sebességszökkentő hatása^{1,2}.

Anyageltávolítás szimulációnál a szerszám által kihalított anyagterefogatot lépésről lépésre generáljuk a szerszám pillanatnyi helyzetének megfelelően. A szerszám teljes moz-

gása során alakul ki az a söpört térfogat, amelynek eltávolítása a készterméket eredményezi. Ha a szimulációs lépések között elég kicsi a szerszám által végzett elmozdulás, a szerszám által söpört térfogat az egymás után következő szerszámtérfogatok uniójaként értelmezhető, egy adott hibahatáron belül. Amennyiben a hibaküszöbön belül maradás csak olyan kicsi elmozdulásokkal biztosítható, ami jelentősen megnöveli a szimulációs lépések számát, így lecsökkenti a szimuláció sebességét, a söpört térfogatot más, hatékonyabb módszerekkel kell számítani. Ehhez leggyakrabban analitikus módszereket használnak, amelyek poligonos (háromszög-alapú) felületreprezentációs térfogatot eredményeznek (lásd pl. itt^{3,4,5}). Ez nem jelenti a teljes söpört térfogat egy lépésben történő generálását: a szerszám pályája lokális sajátosságainak figyelembevételével az egymást követő "könnyebben" kiszámítható szakaszok söpört térfogatának meghatározásával alakul ki a teljes térfogat.

Az általunk kidolgozott módszer sajátossága, hogy a szerszám térfogatát LDNI képek formájában generálja, és az anyageltávolítási számításokat végző algoritmus közvetlenül e képeket használja fel. Az LDNI képek a szerszám felületének egy adott irányból történő rétegekre bontásával alakulnak ki, és a felület pontjainak mélységi értékei mellett az adott ponthoz tartozó normálvektort is tartalmazzák⁶. (La-

tered Depth-Normal Images = rétegekre bontott mélységi-normálvektoros képek). Legegyszerűbb esetben a söpört térfogatot a sűrűn egymás után, depth-peeling eljárással felvett szerszám-képekből nyerjük. Ez lehetővé teszi olyan bonyolult szerszámok bevonását is az anyageltávolítás szimulációk körébe, amelyek analitikus leírása nehézkes, vagy a vágási számítások analitikus módszerrel való elvégzése nem megoldható⁷ (lásd még a hivatkozott⁸ demonstrációs videót). A szerszám-pálya-szakaszok söpört térfogatának LDNI képekben való ábrázolása a szakirodalomban megtalálható módszerek segítségével két lépésben alakul ki: a söpört térfogat felület-reprezentációjának kialakításával és annak LDNI képekbe renderelésével. Ez egyrészt CPU alapú számításokat, másrészt CPU-GPU adatmozgást igényel. Az alábbiakban olyan módszert mutatunk be, amely az LDNI formában leírt söpört térfogatot egyetlen lépésben állítja elő, a grafikus hardver kizárólagos felhasználásával. Ehhez a számítógépes grafikából jól ismert shadow mapping technikához hasonló eljárás került kidolgozásra: a térfogatifelület több irányból felvett mélységi képeinek transzformációjával alakulnak ki az LDNI képek.

2. A GPGPU alapú anyageltávolítás szimulációs eljárás bemutatása

2.1. Technikai háttér

Az egységesített architektúrájú grafikus hardverek néhány éve jelentek meg a piacon. Az egységesített architektúra azt jelenti, hogy a grafikus hardveren futó, különböző feladatokat végző programok (shaderok) az elvégzendő feladattól függetlenül ugyanazon hardveregységeken futnak, nem pedig az egyes feladattípusokhoz külön-külön kifejlesztett dedikált egységeken.

Az egységesített hardver-architektúra lehetővé tette a hagyományos vertex és pixel shaderokon kívül egyéb, akár nem feltétlenül grafikus jellegű shaderok definiálását is. Erre példa az ún. geometry shader, amely geometriai primitíveket (pont, vonal, háromszög) képes kezelni: transzformálni, átalakítani, törölni, és ami az egységesített architektúra megjelenése előtt nem volt lehetséges: a CPU közreműködése nélkül, önállóan létrehozni is. Az objektumok nem feltétlenül geometriai primitíveket jelentenek: egy pont típusú objektum bizonyos attribútumokkal kiegészítve leírhatja például egy gázmolekula térbeli helyzetét és sebességét is, a geometry shader pedig e molekulák mozgási egyenleteinek ciklikus számításával a teljes gázhalmaz mozgásának szimulációját végezheti, adott külső és belső erőhatások figyelembevételével⁹. Az eredményeket - egyéb shaderok bevonásával -, a grafikus hardver meg is jelenítheti.

A programozó tehát kizárólag a grafikus hardver segítségével hozhat létre és manipulálhat objektumokat, teljes egészében átvéve ezeket a funkciókat a CPU-tól. Ennek a megoldásnak az az előnye, hogy a nagymértékben párhuzamosított számítási feladatok elvégzésére kifejlesztett grafikus

hardveren lényegesen gyorsabban végezhető el az SIMD (Single Instruction on Multiple Data) jellegű utasítássorozatok, mint a CPU-n, továbbá a CPU-GPU adatátvitelt minimalizálva csökkenthető a korlátozott adatátviteli sávszélesség futási teljesítményt csökkentő hatása.

Az egységesített architektúrájú grafikus hardverek általános elnevezése a GPGPU (General Purpose Graphics Processing Unit), utalva a nem grafikus jellegű felhasználhatóságra. Az általános célú felhasználást speciális programozási felületek, (ún. API-k, Application Programming Interface) segítik. A geometry shaderen kívül egyéb, nem grafikus jellegű shaderok is megjelentek (pl. compute shader).

2.2. Az eljárás leírása

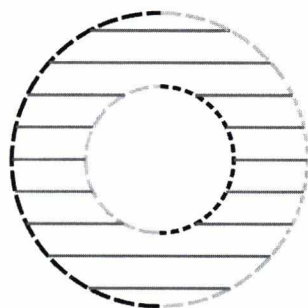
Az általunk kidolgozott eljárás szabad formájú testek anyageltávolítás jellegű megmunkálás-szimulációját végzi. A "szabad formájú" jelző tipikusan valamely CAD rendszerben létrehozott, tetszőleges formájú, manifold (valóságos térbeli) testekre utal. A megmunkálás anyageltávolító számmal történik, amelynek pozíciója és orientációja szabadon és egymással szinkronban változtatható a művelet során. Az eljárás így alkalmas 5 tengelyes marógép által végzett megmunkálás szimulációjára.

Az eljárás során a munkadarab az ún. dixel bázisú térfogatreprezentációs módszer egy továbbfejlesztett változatával kerül leírásra. Az eredeti dixel bázisú módszer a következőkből áll: egy térbeli síkra adott sűrűségű rácsot helyezünk, és minden rácspontból a síkra merőleges egyenest indítunk. A továbbiakban az egyenesek és a munkadarab metszéspontjait vizsgáljuk. Ahol egy egyenes áthatol az anyagon, egy dixel (depth element, mélységi elem) keletkezik, amelyet az anyagba való behatolás, illetve az anyagból való kilépés térbeli koordinátaival, vagy az ezekre visszavezethető értékekkel (pl. a belépés koordinátái, a dixel iránya és hossza), illetve egyéb kiegészítő paraméterekkel (pl. szín) definiálunk. Ha egy egyenes többször is metszi a testet, az egymás után következő dixeletet láncba fűzve tároljuk a memóriában.

Esetünkben a GPU-k speciális adattárolási és -feldolgozási módjaihoz igazodva a dixel alapú reprezentáció az alábbiak szerint történik: A dixeletet nem csak egy, hanem három, egymásra merőleges (ortogonális) irányban vesszük fel (ún. tri-dixel vagy multi dixel módszerrel). A testet az egyes irányokhoz tartozó dixelet teljes egészében leírják, tehát három teljes értékű dixel bázisú leírást képzünk, különböző irányokban. Az ily módon definiált dixeletet vertex bufferekben kerülnek tárolásra a grafikus hardver memóriájában.

A nyers munkadarab (előgyártmány) multi dixel alapú leírását konvertálással képezzük a munkadarab felületét leíró reprezentációból (BRep, Boundary Representation). Ilyen reprezentáció például a felületet háromszögekkel és normálvektorokkal leíró, a sztereo litográfia során alkalmazott

módszer, amely .stl kiterjesztésű fájlokat használ a tárolásra. Esetünkben a konverzió a következőképpen történik: A munkadarabot leíró BRep objektumot a dexelek irányvektorai mentén depth peeling eljárással rétegekre bontjuk. Ez azt jelenti, hogy az objektumot az adott irányból rendereljük, és az így látható felület pontjainak a kamerától való távolságát és a normálvektorát LDNI kép formájában tároljuk, majd ezt követően a látható felület mögötti, felénk irányuló takart felületekről is ilyen képeket készítünk, egészen a leghátsó takart felületdarabkáig. Az eljárást megismételjük az objektum hátoldali felületeire is. A hozzánk legközelebb eső elülső és hátsó felületrészek adják az első réteget, a legtávolabbiak a legutolsót. A dexeleteket az egyes rétegek elülső és hátoldali képei alapján hozzuk létre: a dexeletek behatolási és kilépési koordinátái az elülső és hátsó kép pontjainak kép- és mélységi koordinátái alapján számíthatók a kamera-térből világ-koordináta-rendszerbe való transzformálással, a dexeletek normálvektorai pedig az azonos képpernyő-koordinátákhoz elmentett normálvektorok lesznek. (Lásd: 1. ábra.) A konverziót mindhárom dixel-irányban elvégezzük.



- Layer 1, front face
- Layer 1, back face
- Layer 2, front face
- Layer 2, back face
- Dexels

1. ábra. A dexeletek létrehozása.

Az anyageltávolítás szimulációja során a munkadarabba behatoló vágószerszám térfogatát vagy az általa a mozgás egyes szakaszain végigsöpört térfogatot a szerszám mozgáspályája mentén lépésről lépésre eltávolítjuk a munkadarab térfogatából. Ez a dixel alapú térfogatreprezentáció esetében azt jelenti, hogy minden egyes dixelre meg kell határozni a szerszám és dixel metszéspontjait, és el kell távolítani a dixelből kivágott részeket. Egy metszéspont esetén a dixel rövidebb lesz, két vagy több metszéspont esetén a dixelből több dixel keletkezik. A műveletet esetünkben a Geometry shader végzi, két lehetséges módon:

- Koordináta-geometriai számításokkal. A dixel, mint irányított térbeli egyenes-szakasz, és a vágószerszám alakját leíró térbeli felületek vagy térfogatok metszéspontjainak analitikus számításával.

- Depth peeling eljárás segítségével, ahol a szerszámot a

dexeletek létrehozását leíró részben vázolt módszerrel, a dexeletek irányvektorai mentén rétegekre bontjuk, és az egyes rétegek által definiált térfogatrészeket vonjuk ki a dexeletekből. A kivonás a dexeletek koordinátái és a rétegeket leíró képekből kinyert koordináták összehasonlítása alapján történik. A pontosság növelése érdekében a dexeletek kezdeti létrehozásához és a szerszámok rétegekre bontásához ugyanazon virtuális kamerapozícióból és kamera-paraméterekkel végzett depth peeling eljárást használunk.

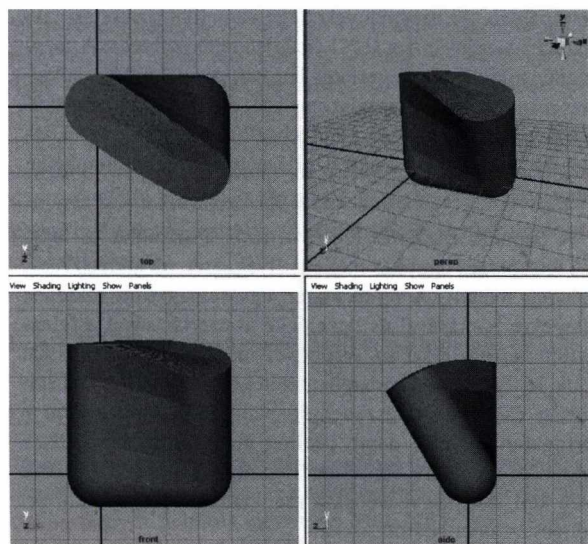
A térfogat-kivonási számításokat a geometry shader ciklikusan, mindaddig végzi, amíg a szerszám végighalad a szerszám pályán. Az eredményeket egy vagy több számítási ciklus után megjeleníti a GPU (megjelenítési ciklus). A megjelenítési ciklusban a grafikus hardver megjelenítésre alkalmas primitívekből (háromszögekből) álló felületekké alakítja a dexeletek összességét, majd a beállított fényviszonyoknak megfelelően árnyalja és kirajzolja az így felépített munkadarabot.

A bemutatott eljárás lehetővé teszi a szimulációs lépések tetszőleges nagyítás melletti elvégzését, megnövelt részletesség mellett. A munkadarab egy felnagyított részének rétegekre bontásával és dexeletekké konvertálásával a felnagyított térfogatrészt az eredetivel megegyező felbontással írjuk le, így a munkadarab méretéhez viszonyított ráctávolság csökken, a részletesség pedig nő. Amennyiben a térfogat-eltávolítást depth peeling módszerrel végezzük, a szerszám rétegekre bontásánál használt kamera-beállítások meg kell, hogy egyezzenek a munkadarab nagyításakor érvényes kamera-beállításokkal.

3. LDNI alapú söpört térfogat generálása

A söpört térfogat generálásának legnagyobb nehézsége az, hogy bonyolult szerszám pályája mentén kell egy olyan zárt, poligon-alapú felületet előállítani, amelyben nem fordulnak elő felület-átmetszések vagy egyéb geometriai hibák. Ehhez lépésről lépésre meg kell határozni a söpört felület kialakításában résztvevő szerszám-felületszegmenseket, és ezek alapján egy teljesen új objektumot kell kreálni. Ez komoly matematikai apparátust igényel, az elért eredmények pedig korlátozottak: az ismert CAM rendszerek 5 tengelyes marási megmunkálások esetén nem is alkalmazzák. (5 tengely szinkronizált vezérlésével hozhatók létre szoborszerű felületek. Söpörési térfogat előállítását 3+2 tengelyig használnak sikerrel; ez esetben a három illetve a másik kettő tengely együttes szinkronizálása nem megoldott.) Mindazonáltal a kapott felület nem nevezhető túlságosan bonyolultnak: a 2. ábrára pillantva láthatjuk, hogy az általánosan használt marószerszámok söpörési térfogata bármely irányból legfeljebb 1-2 takart felületréteggel rendelkezik az analitikusan feldolgozható hosszúságú szerszám pályá-szakaszok esetében. Fontosabb tulajdonság viszont, hogy ezekben az esetekben nincs olyan felületpont, amely ne látszódná legalább egy olyan képen, amelyek a három ortogonális koordinátatengely irányából kerültek renderelésre (elülső és

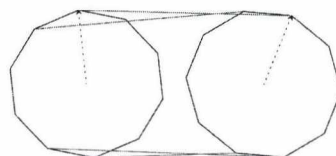
hátsó felületeket tekintve) - tehát ezek a mélységi képek a felületet leíró összes információt tartalmazzák. Az újonnan kidolgozott térfogat-generálási módszer e mellett azon tény felismerésén alapul, hogy az LDNI képek előállításához nem feltétlenül szükséges geometriai hibáktól mentes felülettel leírt térfogat-objektum. A söpört térfogat előállításában résztvevő szerszám-felületszegmensek mélységi képekbe való renderelésénél ugyanis nem okoznak problémát az egymást metsző háromszög-lapok, mivel azok takart (így a söpört térfogat felületén belül eső) részei a mélységi vizsgálat során eldobásra kerülnek.



2. ábra. Marószerszám söpört térfogata.

A kidolgozott eljárás a fentieknek megfelelően a következő: A szerszám háromszög-alapú BRep leírása vertex pufferekben kerül eltárolásra, a hagyományos képalkotási módszereknek megfelelően. A csúcsponti információkon kívül eltárolásra kerülnek a háromszögek között szomszédsági információk is, amint ezt a 4.0-ás illetve magasabb verziószámú shader modellek lehetővé teszik. A szerszám-pálya-szakaszhoz tartozó szerszám-koordináták és orientációs vektorok ismeretében meghatározható a csúcspontok mozgása, így a szomszédsági információk bevonásával meghatározhatóak a söpörési térfogat határfelületének képzésében résztvevő élek (lásd 3. ábra). Ezen élek egymás után következő helyzetéből a geometry shader segítségével alakíthatók ki a határfelületet alkotó (ám egymással akár metszésben levő, tehát nem manifold felületet képező) lapok, amelyek a multi dixel reprezentációnak megfelelő három ortogonális irányból illetve ezekhez képes szemből renderelésre kerülnek (összesen hat képet alkotva).

Az LDNI képek első rétegét a fentiekben kialakított három ortogonális elülső kép adja. A többi réteghez használjuk fel azt a tulajdonságot, amely szerint az összes felület-pont megjelenik legalább egy képen a hatból. Azon pontok, amelyek az elülső képen, tehát az első rétegen nem szerepelnek,



3. ábra. A söpört térfogatban résztvevő élek meghatározása (2D).

de valamelyik másikon igen, takart felületen, tehát valamelyik hátsóbb rétegen helyezkednek el. Ezek alapján a hátsóbb rétegek előállításánál a következő módszert követjük: a határoló lapokat újra renderelve a pixel shader szakaszban elvégezzük a felületpontoknak a hat LDNI képre történő projekcióját, és amennyiben az valamely másik képen szerepel (vagyis a kép- és mélységi koordinátái megegyeznek), de a renderelési irányhoz tartozó képen nem, takart felület-ponttal állunk szemben. A képernyőpontokra mélységi tesztet végezve, az előző rétegek pontjainak eldobásával, rétegenként 1 renderelési ciklussal megkapjuk a söpört térfogat összes LDNI képét.

4. Eredmények

Amennyiben a söpört térfogatot a szerszám-pálya mentén összegzett szerszám-térfogatok adják, szimulációs ciklusonként a szerszám LDNI képeinek előállítása (a vizsgált egyszerű marószerszámok esetén 1-1 réteg a három ortogonális irányból), valamint a dexelek és a szerszám metszéspontjainak számítása történik. LDNI-alapú söpört térfogat előállítása esetén az adott szerszám-pálya-szakasz végpontjaira a szerszám teljes felületének, a köztes pontokra csak a söpörési térfogat kialakításában résztvevő felületszegmenseknek (amely egy nagyságrenddel kevesebb elemet jelent) renderelése történik, jellemzően két réteg kialakításával, továbbá szakaszonként egyetlen metszéspont-számítás a dexelek és a söpört térfogat között. A mérési eredmények szerint a szerszámot vagy söpört térfogat-felületet alkotó háromszögek számának alakulása nem befolyásolja döntő mértékben az eljárás sebességét. Az LDNI rétegek számának megduplázása értelemszerűen megduplázza a depth peeling eljárásra fordított időt. A söpört térfogat előállításának előnye egyrészt abból adódik, hogy a metszéspont-számítások száma drasztikusan csökken: a hivatkozott ^{10,11} demonstrációs videókon látható megmunkálás-szimuláció nagyjából 230 ezer lépésből áll, míg söpört térfogat számítás esetén - a viszonylag egyenletes mozgásnak köszönhetően -, egyetlen szerszám-pálya-szakasz akár száz szerszám-pozíciót is magába foglalhat. Az eljárás másik előnye az, hogy a kicsiny mértékű szerszám-pozíció-változás akár egy nagyságrenddel is megnövelhető a söpört térfogat megnövekedett pontossága miatt.

5. Összefoglalás

A Pécsi Tudományegyetem Pollack Mihály Műszaki Karának Műszaki Informatika Tanszékén évek óta folynak az általános célú grafikus hardverek gépészeti szimulációkhoz történő felhasználására irányuló kutatások. Kidolgozásra került egy GPGPU alapú anyageltávolítás szimulációs eljárás, amely kihasználja az új típusú hardverekben rejlő képességeket: a nagyfokú párhuzamosítás lehetőségét és az egy egységen belül történő feladat-végrehajtás előnyeit. Az eljárás multi dixel alapú objektum reprezentáción alapul, és egyedi, LDNI alapú szerszám-reprezentációt alkalmaz a vágási számítások elvégzésére. Ehhez az eljáráshoz került kifejlesztésre a bemutatott söprési térfogat generáló módszer, amely illeszkedik az alapprojekt GPU-központú megoldásaihoz, miközben tovább javítja annak hatásfokát.

11. <http://www.youtube.com/watch?v=zTcboSFxa44>

References

1. B. Tukora, T. Szalay. High performance computing on graphics processing units. *Pollack Periodica*,**3**(2):27–34. (2008) 1
2. B. Tukora, T. Szalay. Manufacturing simulation in the light of the recent GPU architectures. *Proceedings of Gépészet Konferencia*, (electronical issue): C20 1
3. W.P. Wang, K.K. Wang. Geometric modeling for swept volume of moving solids. *IEEE Computer Graphics Applications*,**6**(12):8–17 (1986) 1
4. Y.C. Chung, J.W. Park, H. Shin, B.K. Choi. Modeling the surface swept by a generalized cutter for NC-verification. *Computer Aided Design*,**30**(8):587–593 (1998) 1
5. K. Weinert, S. Du, P. Damm, M. Stautner. Swept volume generation for the simulation of machining processes. *International Journal of Machine Tools Manufacture*, (44):617–628 (2004) 1
6. Y. Chen, C. L. Wang. Layered Depth-Normal Images for Complex Geometries - Part One: Accurate Modeling and Adaptive Sampling. *ASME Computers and Information in Engineering Conference*, DETC2008-49432 (2008) 1
7. B. Tukora, T. Szalay. Fully GPU-based volume representation and material removal simulation of free-form objects. *Innovative Developments in Design and Manufacturing: Advanced research in virtual and rapid prototyping*, pp. 609-614. (2009) 2
8. <http://www.youtube.com/watch?v=t2XLnpgrGP4> 2
9. W. Liu, B. Schmidt, G. Voss, W. Müller-Wittig. Molecular dynamics simulations on commodity GPUs with CUDA. *Lecture Notes in Computer Science*, Vol. 4873/2007, pp. 185-196. 2
10. <http://www.youtube.com/watch?v=TwPgrgCgsX4> 4

Immersive Interactive Film & Real-Time Computer Graphics

Barnabás Takács,^{1,5} Gábor Szijártó¹, Gergely Komáromy-Mészáros¹, Dávid Hanák², Levente Dobson³ and Zoltán Kömíves^{4†}

¹ Technical University of Budapest, Hungary.

² Nav N Go, Budapest, Hungary.

³ Sense/Net Zrt, Budapest, Hungary.

⁴ Royal Scottish Academy of Music and Drama '12, Glasgow, United Kingdom.

⁵ Digital Elite Inc., Los Angeles, CA, USA.

Abstract

This paper describes the methodology of a novel entertainment experience that combines the benefit of movie quality film production with interactive capabilities available on the DVD version or over the Internet. Our system is based on Panoramic Broadcasting (PanoCAST) that utilizes spherical video to capture the entire scene and subsequently turn this footage into an interactive experience with the help of tracking and advanced interaction modalities, including virtual-reality-on-demand. We present a case study on a short film based on Franz Kafka's Metamorphosis.

Keywords: Panoramic Broadcasting (PanoCAST), Interactive Film, Immersive Media, Virtual Reality On-Demand, Virtual Human Interface.

1. Introduction

Entertainment technology, in recent years, has been increasingly dominated by new forms of interactive experiences, such as games, that harness the power of computing power and place users in a world they control themselves. From the perspective of entertainment computing this global phenomena shifted high-tech interests away from traditional media, most notably film, and created a new wave of opportunities that focus on *shared experiences* and as a result on *social media*. To keep up its hi-tech appeal and unsurpassed visual and production quality the film industry invested in digital cinema² and most recently 3D movies. While these steps successfully brought audiences back to movie theatres around the world, the need to combine film, classically considered as a passive experience, with interaction still remains a challenge.

We propose herein a methodology that combines the benefit of movie quality production that result in a film projectable in theatres, with interactive capabilities invoked on the DVD version or over the Internet. Our system is based

on *Panoramic Broadcasting* (PanoCAST) that during production captures a *spherical video* of the entire scene and delivers the above mentioned outputs simultaneously as the result of the same production process.

The remainder of this paper is organized as follows: First we briefly review panoramic broadcasting architecture our current solution is based upon. Next we discuss how to bring interaction to the realm of film without hurting the interest of artistic expression followed by a description of advanced interaction modalities. The methodology to turning panoramic film into a truly interactive experience using tracking and clickable content is followed by a brief case study and our conclusion.

2. PanoCAST - A Novel Architecture for Interactive Entertainment

PanoCAST or Panoramic Broadcasting⁵ is a technology originally designed to deliver telepresence-based entertainment services over the Internet or mobile networks and thereby allow viewers to experience the very feeling of being somewhere else from their physical location⁹. The basic concept of the architecture is as follows: We first capture a

† Part of this work was performed while at MTA SZTAKI

stream of high fidelity spherical video (often called immersive media) with the help of a special camera system with six lenses packed into a tiny head-unit⁴. The images captured by the camera head are compressed and sent to our server computer in real-time delivering up to 30 frames per second, where they are mapped onto a corresponding sphere for visualization. The basic recording and server architecture, then employs a number of virtual cameras and assigns them to each viewer thereby creating their own, personal view of the events the camera is capturing or has recorded. The motion of the virtual cameras is controllable via TCP/IP with the help of a script interface that assigns camera motion and pre-programmed actions to key codes on the mobile device. The host computer then generates the virtual views each user sees and streams this information back to their location using RTSP protocol. This concept is demonstrated in Figure 1 in the context of a rock concert. The spherical camera head (left) is placed at the remote site in an event where the user wishes to participate. The camera head captures the entire spherical surroundings of the camera with resolutions up to 3K by 1.5K pixels and adjustable frame rates of maximum 30 frames per second (fps). These images are compressed in real-time and transmitted to a remote computer over G-bit Ethernet connection or using the Internet, which decompresses the data stream and re-maps the spherical imagery onto each viewer's display device locally. The key idea behind the *PanoCAST* architecture, as opposed to just simply streaming the entire spherical video, is to distribute "just-in-time" views created by the virtual cameras. This forms the foundation of the proposed interactive film technology as discussed in the following sections.

3. Film vs. Interactive Experience - Retaining Artistic Intent

For many directors, photographers and camera crew, *interaction* is a word that goes against the artistic intention of the creators. Specifically, the very concept of a frame that can guide our eyes to bring our attention to the smallest details as well as showing sweeping overall views that spur engulfing emotions, allow artists to include what is of momentary interest and exclude everything else. By this definition reality is "boring" and artistic talent is precisely what allows us to focus us on what is truly important. If we simply allowed viewers to look around (interactively) how would this be possible?

To answer this challenge one is required to bring *interaction* into the picture without interfering with the artistic process itself while still serving the underlying message of the creators themselves. To achieve this goal we set up a two-stage production where in the first leg we record the entire film sequence using a panoramic video head but used almost as if it was a regular film camera aimed directly at the scene. The camera crew watches the output of one of the six camera modules to see the composition and the action, but in the

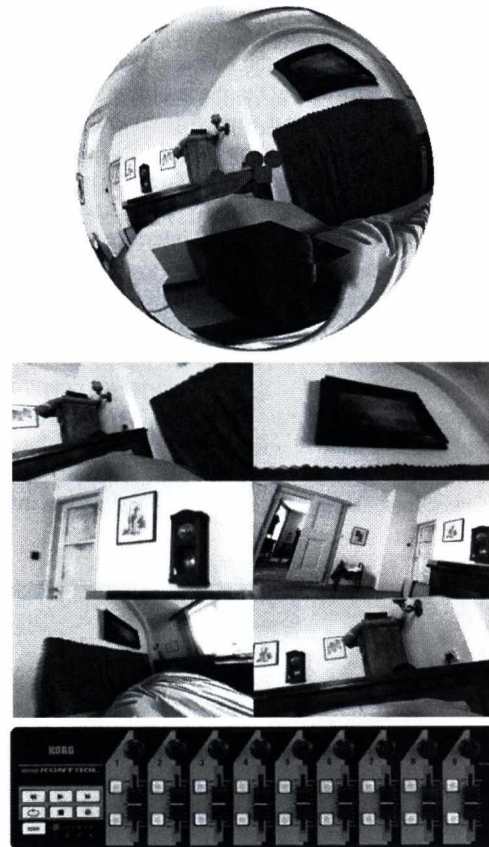


Figure 1: The *PanoCAST* pre-visualization and editing system uses MIDI-channels to control a set of virtual cameras with the help of physical controls. Upper: The spherical image mapped shown from outside with the virtual camera in the center. Center: Six independent camera views of the same scene shown on the left. Below: KORG Nano Kontrol used to direct multiple virtual cameras and edit key frames.

second leg, during post production, we sit down with them again to create a second pass using virtual cameras placed inside the spherical film itself. It is at this stage where the final shape and look of the film takes place. By aiming the virtual camera at different portions of the spherical recording, changing the field of view and moving the virtual camera the director decide upon the final imagery that will eventually be rendered in high resolution (4Kx4K frames) to be finished in the film lab and printed on film. Due to the spherical nature of the recording, however, if needed the field of view may be extra large thereby introducing new visual ways of representing scenery or the emotional as well as the mental state of a character. The result is stunning imagery that is iconic and instantly recognizable, yet it can be projected in theatres using the traditional channels of film distribution.

During a film shoot the neither the director nor any mem-

ber of the staff are present on the scene. Only a remote controlled robot moves the camera and interacts with the actors. Thus, to see how good the take was, on-set the director uses a virtual reality HMD to place himself in the scene and to see if he/she got what wanted from the actors. This pre-visualization tool is based on the Virtual Human Interface (VHI) module of the system¹. Off-set, during post production, the director works with physical camera controls (*KORG Nano Kontrol*) that use *MIDI* interface to communicate camera commands to the rendering system (see Figure 1). This allows natural interaction and quick review of the different visual possibilities offered by the *PanoCAST* system. While in most cases the panoramic camera was moving constantly on the set with the help of a remote controlled robotic camera (see sections below), in some instances the entire output of the spherical lenses was used to produce the final shot. This is demonstrated in Figure 2 showing how the final scene, called “bubble sequence” in the film was made.

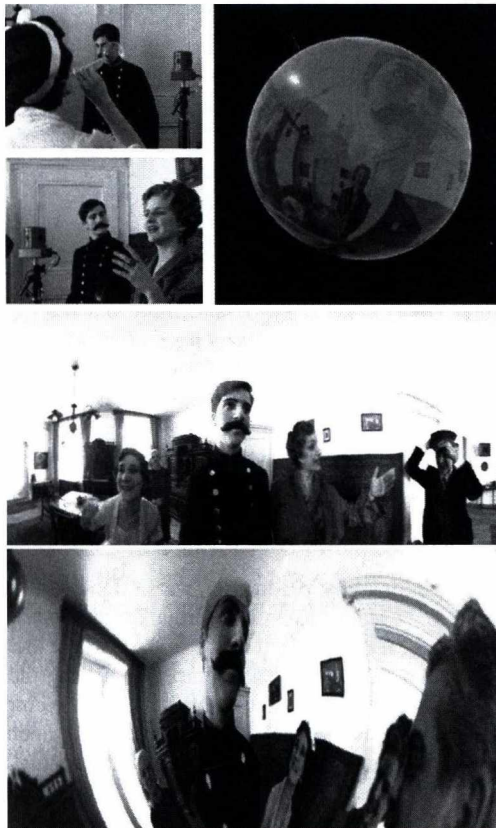


Figure 2: Creating the “bubble sequence”. A static camera surrounded by actors (upper left) was used to create a stretched full circle panorama image (center) and subsequently mapped onto the surface of a deforming sphere through the wall of which the camera passes (below) to reveal the soap bubble floating in space (upper right).

To create an immersive interactive experience using the same panoramic footage that complements the film and provides additional insight we created a spherical video player that may run on a computer installed locally or accessible via the Internet³. In this mode viewers are allowed to control camera parameters interactively and individually thereby exploring the content of the film in a different manner. However, to represent the original concept and the artistic intention of the creators the same *virtual camera track* used for rendering the movie version is retained in the interactive mode as the default behavior of the viewer. Thus, in passive mode, i.e. when viewers do not want to take control they see exactly how the film was originally meant to be seen, yet when they decide to turn their attention to somewhere else, they can literally “enter that world” and feel as if they are in the very center of the action⁵.

4. Using Real-Time Computer Graphics to Create Spherical Visual Effects

To turn the original panoramic footage into the scenes requested by the director, we used computer graphics techniques to map the spherical space onto the image plane. As an example, many times the director and/or the viewer needs to view entire panoramic image on one stretched screen. In this case the full panoramic image may be mapped onto a plane, where the X axis will be line of latitude, while the Y Axis will be line of longitude, much similar to the process of creating a flat map (Figure 3). This method is called in cartography: the Plate Carrée (“plane square”). After this transformation the lines that are straight in the real world will not be straight lines, but rather lie on a sinusoid. Another problem was that instead of a true spherical coordinate system, in fact our camera module produces overlapping patches (see also Figure 4). Thus, for our case, when using this transformation we had to pay attention special attention not to introduce artifacts that disturb the visual results.

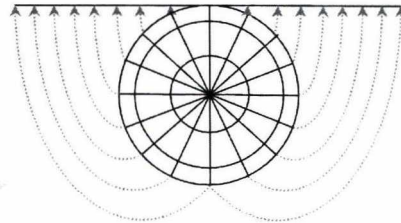


Figure 3: Basic configuration of mapping a spherical image onto a plane.

As mentioned above our panoramic system contains 6 cameras, thus 6 separate images are received from the recording system, each mapped onto a different image patch. When we want to assemble these images to a full panoramic image, we must solve some basic problems. First, as the focal point of each camera is not the same point in 3D space

(i.e. having a larger set of overlapping fields of view to avoid dead space), the compositing of requires blending. As we have no prior information on distances, objects with an infinite distance from the camera would have sharp and perfectly matched images in the overlapping areas. However, in real life, the images on each camera patch do not match correctly due this different focus point. This is eliminated by edge-blending to reduce sharp artifacts. The second problem is internal representation. The raw data from the camera head is a set of 6 separate image streams. If only the combined "stretched" image is stored once the above defined mapping took place, in other words the neighbors of any pixel are only those pixels, which are next to it on the final space. On the other hand, this encoding has a huge drawback as well; namely that the pixel density is not balanced. At the poles the density is much higher than on the equator. Of course, if the texture is to be used only in stretched mode it will not make any difference. But when it is used on a zoomed sphere it will cause visible artifacts - longish looking pixels near the poles. Moreover the storage from a graphics processing point of view is not ideal due to this unbalanced pixel density. Avoiding the pixel density problem, the stored image may be stored as a cube map. The pixel density will be well balanced. However, in this case there is another problem: artifact on the edges of cube, if the images are stored in any compressed format. We use long timing videos, thus the compression is necessary. The compressor has no knowledge of any information regarding pixels on the edges, therefore it creates an unwanted line at the edges of cube. To conclude, we found that the best way to store the images, was to keep the original six raw camera images and composit them as a 2x3 layout. During rendering these images are blended, without any artifact. The pixel density is also balanced.

Based on the above arguments in our visualization system the composition of the final spherically transformed image happens in real-time at the rendering phase. If we want to use only the stretched image, the final image can be rendered with a simple pixel shader. In this case the rendering is essentially a ray trace algorithm. To implement this functionality we need only a look up table encoding camera identifiers (IDs) visible for a given direction. This table can be stored in a cube map, thus the camera IDs can be readily accessed. Clearly, in any one direction there can be a maximum of three camera images. For blending them we also store there the alpha values for each camera. Thus, each pixel of this cube map contains 3 UV coordinates and one alpha value. UVs must be float at least two 16bit one, while alpha can be 8 bit integer. The best case stores one pixel of the look up table in 9 bytes. As the cube map resolution must be high, this requires rather large storage and heavy on resources. Our aim is that this system can be executed on average computer. Another way to render the panoramic image is using the geometry itself. The solution requires only the geometric representation of the camera patches and an alpha map that contains the blending information we need. The

resolution of this geometry is very important for transformations and producing a high quality final image. As mentioned the flattening or "stretch" algorithm transforms straight lines onto curves. If the resolution of the core geometry is not high enough, after the transformation the lines will easily become fragmented. On the other hand this solution is very simple and the resolution can be easily decrease for lower performance computers.

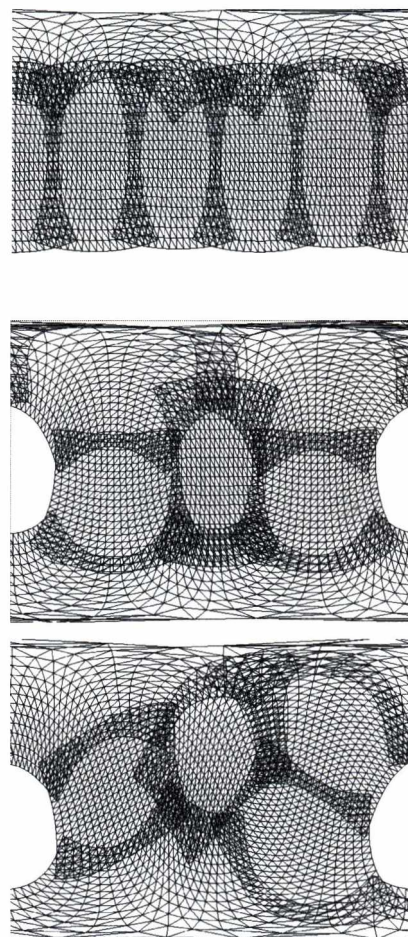


Figure 4: Different configurations of the flattened/stretched sphere showing the individual patches in wire frame mode. (Top - Sideway, Center - Upright, Bottom - General direction)

The above briefly introduced "final" stretching algorithm we used in this film is based on a transformation from 3D polar coordinate system to 2D. This is illustrated in Figure 4. The six camera overlapping patches of the camera images are shown in different configurations (Sideway, Upright, and General Direction). The white circular area corresponds to the minimal-size "blind spot" of the imaging system. A critical advantage of this mapping is that it does not explicitly

take advantage of the geometric shape, as it transforms triangles into other triangles. Therefore the pole and latitude cutting edge can be easily changed via a 3D transformation matrix on the geometry. Each vertex is transformed using this matrix and once the longitude and latitude values are calculated it is displayed as part of the final image. The only rendering problem involves those triangles at the cutting edge and the poles that fall apart. This simply happens because some of its vertices fall on one side, while the rest of them to the opposite side. Yet, these triangles are easily detected. In our application there are many triangles, which are split by this cutting edge. To detect them we use the observation that in the stretched domain the latitude of a triangle should not be more than 180 degrees. Furthermore 180 degree value can only occur only the case if the triangle contains the center point of the transformation. Thus, these split triangles are detectable using the latitude values. Having detected these degenerate triangles we create two independent polygons along the cutting line. In most cases this results in one triangle and a quad, thus there will be 2 additional triangles. On the other hand to find the new vertices and to detect the special cases causes computational overhead that may slow the rendering system down. To avoid this critical bottleneck we found yet a simpler way, by duplicating the cutting triangle and placing it both sides simultaneously (i.e. there will be one at the left side, one at the right side). The vertex, which is on the other side, is added or subtracted 360 degree to lateral to get the required vertex. As a result, without any complex calculation the transformed image will be filled automatically and the redundant part (more than 360 degrees) is eliminated when rendering the final image on the rectangular screen.

5. Advanced Modes of Interaction - Virtual Reality on Demand

The PanoCAST system offers several interaction modalities, the first one of which is the viewers' ability to freely look around. While the film rolls the motion of a virtual camera assigned to them may be controlled by each viewer independently using the mouse, the keyboard or even advanced game controllers, like the Wii or tracked orientation data obtained from a *head mounted display* (HMD). It is this later mode that virtually allows users to enter the film and feel almost as if they were there together with the actors. The online version of our system has been implemented in Adobe Flash. To handle input streams from a variety of I/O streams we created a tiny program (called *VHI DeviceManager*) that upon installation connects to all available physical sensors (see webpage for a list of supported devices⁵) and represents this information to the player for interactive purposes. The simplest of this interaction is change in head orientation (i.e. looking around). More complex rules are defined in a *dynamics* descriptor file that defines the onset and offset of *triggers* via simple sets of rules and applies these rules to the scene accordingly. Triggers may detect simple click

over a region of interest in the image (see section on *Clickable Content*) start events and even physical simulation to seamlessly integrate animated 3D models into the original recording.

Therefore we argue that the infrastructure that forms the basis of on-demand virtual reality and novel forms of interactive entertainment over the Internet as well as on mobile platforms already exists and may be utilized to create *interactive films* (Figure 5). Specifically, with the advent of low-cost computer accessories, peripheral devices and even HMDs with built in trackers that cost below \$500¹⁰ inspired by and deployed in the use of VR offers more intuitive interfaces and applications areas than ever before⁷⁸. Thus, our *Panoramic Broadcasting* system combines three key elements which, a new form of interactive film that advances the state-of-the-art and radically changes the way we entertain ourselves and consume media. We extend the boundaries of video-on-demand to *Immersive Interactive Reality* (IIR) where not only the content, but also the personalized point of view of the user may be changed interactively. Moreover, we move away from 3D models and traditional VR and use i) *spherical film* in combination with ii) tracking technology to create *Clickable Content*, and iii) *physical simulation* for advanced interaction. This is the subject of the following section.

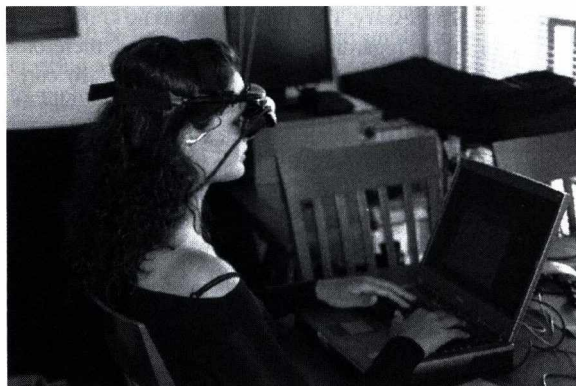


Figure 5: User experiencing full VR immersive interactive reality programming over the Internet using our PanoCAST player, VHI DeviceManager, and built in Physics Engine implemented in Adobe Flash.

6. Clickable Content and Tracking Interface

To make interactive film more exciting and appealing to large audiences a set of extra features needed to be developed that offer more than just the mere ability to look around and being in the center of the action. First, we wanted to allow viewers to learn more about the film itself, being able to click on actors and recall the role of that character, the plot itself or any other linked information that may be of value

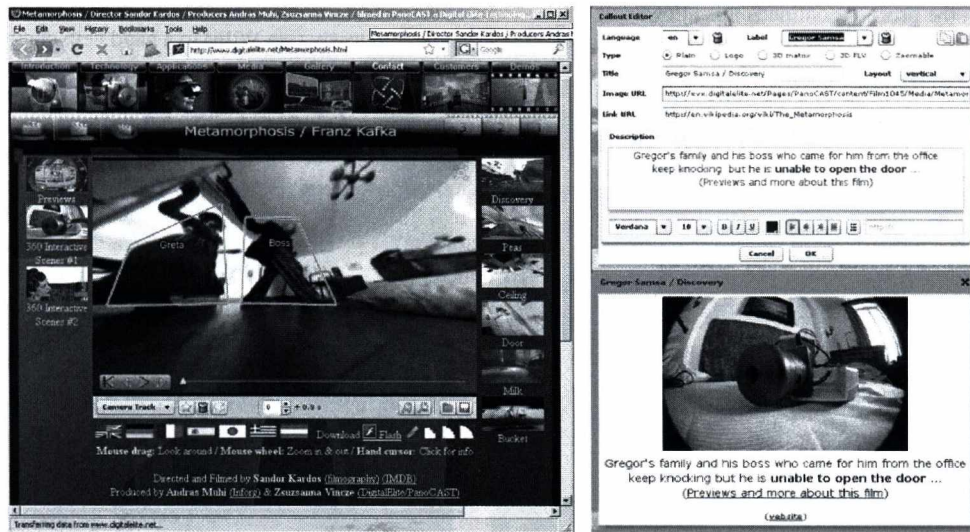


Figure 6: Left: On-line tracking interface to create clickable content and manage callouts in multiple languages. Right: Callout editor interface, below Callout as shown in the application.

and interest. To do this, we developed a pipeline to produce what we call *Clickable Content* (CC).

In simple terms CC means that whenever the user clicks on the scene the rendering engine “fires a search ray” from the viewing camera and orders each visual elements as a function of their distance along this ray to find the closest one. Then the algorithm returns the object’s name, the exact polygon it is intersected at, and the texture coordinates of the intersection point itself. This information allows the system to assign high level actions, such as displaying a callout image and/or text message when clicking on an object of interest in the interactive film or alternatively to directing the user to a website where further information may be found. For easier handling individual texture coordinates can be grouped by regions and tracking algorithms are used to annotate each element of the scene during a broadcast. The final output of such algorithms is a set of tracked regions with actions, visual and text information and web pages assigned on each of which *PanoCAST* viewers can click on during a film show and learn more about the event or simply educate themselves.

Tracking may be difficult as lighting often changes drastically, thus features do not necessarily retain their low level image characteristics. In addition, tracking on the live plate is also hampered by the slow frame rate (25 fps for film) and the fast movements characteristic of the particular film we at hand. Furthermore, since both the Panoramic camera as well as the actors may move in any given scene, we needed to combine *automated tracking* to insert key frames and filter the motion of touch-sensitive panels with post-production tools to correct the output of these algorithms ⁶.

Low-level tracking was implemented using a variety of algorithms including region-based tracking algorithms, such as normalized correlation, texture-based trackers, and optical flow-based estimators. As mentioned above, due to the lack of stable image context low-level trackers exhibit serious limitations in a real-world environment therefore the use of higher level implicit structural models always requires for accurate registration. In our case, what makes this process possible is the large tolerance to errors, as our goal is not to track the tiny details of each actor, but rather to create panels (generalized rectangles) that cover large areas and move along with the actors. So to overcome these problems a hybrid tracking algorithm is used to combine the advantages of each low-level method while minimizing the overall sensitivity to noise and variations in imaging conditions. We start with a *Point Tracker* and adaptively select the best method for each tracking point in a given frame in the region selected by an operator. The image is then processed multiple times and the results are integrated, minimizing the error and creating a smooth transition.

One advantage of tracking in a panoramic video is the system’s ability to keep objects of interest always in the center of the camera. In other words, when following a single point in the scene, the context and window size in which the tracking algorithm operates is kept constant by moving the virtual camera’s target proportionally to the displacement measured on that feature. In other words, the when the raw tracking algorithm outputs a displacement in pixel space, that information is mapped onto the 3D geometry of the spherical video and its relation to the virtual cameras relative geometric arrangement and the target of the camera is displaced such that the feature being track remains is brought back to the center

of the image. As the process continues this allows our solution to employ larger than usual tracking windows without explicitly losing context around the edge of an image.

The output of this process is a set of “invisible” panels the viewer may click on to get access to more information. Tracked panels are *named* and as such they may also be selected from a list and used to follow any given person or object with the help of the virtual camera. When selecting this mode viewers may enjoy looking always at the same actor or actress irrespective of the original camera track. Finally, we briefly mention that the Internet version of our system supports a multitude of Community features as well, thereby allowing on-line users to discuss and share their experiences. As an example, a *Chat* interface was created based on the scene the viewers are looking at and general statistics on camera moves, clicks and interaction data is collected anonymously for subsequent analysis and user characterization. By recording this information it also enables the system to generate a novel camera track that reflects what the majority of on-line viewers were most interested in to look at. This voting scheme or similar alternatives may also be used in large venues during interactive presentations of the film itself, thus maintaining the original concept of the artists but bringing an element of audience participation in as well.

Figure 6 shows a screenshot of the *tracking interface* and *callout editor* in on-line mode. With the help of these tools production staff may track visual elements on-line, review and edit keyframes and assign callout information in multiple languages. When viewers view the same scene, a tiny hand appears discretely above clickable areas encouraging them to learn more if interested. Once these tracks are added they may be referred by name and viewers can direct their virtual camera to follow e.g. “Greta” in all scenes. For more details the interested reader is referred to ³ or for a similar interactive cultural application ¹¹.

7. Kafka's METAMORPHOSIS - A Case Study

To validate our concept of interactive Immersive Interactive Film we produced a short film based on Franz Kafka's famous story, *Metamorphosis* ³. The original novel tells the story of a traveling salesman who one day waking up finds himself turned into an insect ¹². While the spherical nature of the camera head is ideally suited to the subject of the film, the key challenge we first faced was to maintain an extra large field of view while excluding everything unwanted in the picture. More specifically, because panoramic video records and maps the entire 360 scene with only a minimal blind spot, production crew, lighting and stuff, even the director needed to remain hidden entirely.

To achieve maximal visual coverage and also to create an extremely low-angle view of the set as envisioned from the insect's first person perspective, we devised a remote controlled robotic camera platform that was directed and programmed to interact with the actors. This is demonstrated in

Figure 7. The recording system (shown on the left) was connected wirelessly (via WiFi) to a hand help computer (*ASUS R2H Ultra Mobile PC*) that in turn controlled the motors and sensors of an *iRobot Create* platform the camera head was mounted on (right). Using the UMPC's touch screen driving the robot became a simple task. We also enabled the camera system with macro capabilities, frequently used to “teach” the robot a series of actions to carry out without human intervention. As an additional convenience we also included a Wii controller, that was connected to the UMPC and the robot via a Bluetooth link, the accelerometers and programmable buttons of which offered further convenience. As an example, by simply tilting the *Wii* in one or another direction, the robot and thus the camera would start to move. The key features of this programmable camera interface are shown in Figure 8.

8. Conclusion

The powerful graphics capabilities of portable devices in combination with and high-bandwidth connectivity over the Internet provide a novel opportunity for new forms of interactive entertainment in general, and virtual reality in particular. In this paper we described a novel content production and distribution architecture for Immersive Interactive Film, called *PanoCAST*, that is capable of simultaneously producing a traditional film sequence and turn that same footage into an interactive experience for DVD or on-line viewing. The system employs panoramic video as a starting point and create clickable content with the help of feature tracking to turn every element of a panoramic film scene into a link for further information. Computer Graphics tools were used to create special spherical effects that represent a new and iconic (i.e. instantly recognizable) visual quality to the entire film sequence. We discussed the details of this architecture and demonstrated how it is used in the context of a film project. Based on our results and the production experience we argue that an interactive film and as it relates to virtual reality represents a viable path to produce engaging and immersive media content and as such it may serve as the foundation of new generation media.

References

1. Digital Elite Inc.
<http://www.digitalElite.US.com> 2009. 3
2. Charles S. Swartz, *Understanding Digital Cinema: A Professional Handbook*, Elsevier. 2004. 1
3. Immersive Interactive Film of Franz Kafka's *Metamorphosis*,
<http://www.digitalelite.net/Metamorphosis.html> 3, 7
4. Point Grey Research, *LadyBug2 Camera*,
<http://www.ptgrey.com/products/ladybug2> 2009. 2



Figure 7: Remote controlled robot camera used to record the panoramic videos for the Metamorphosis (see text for details).

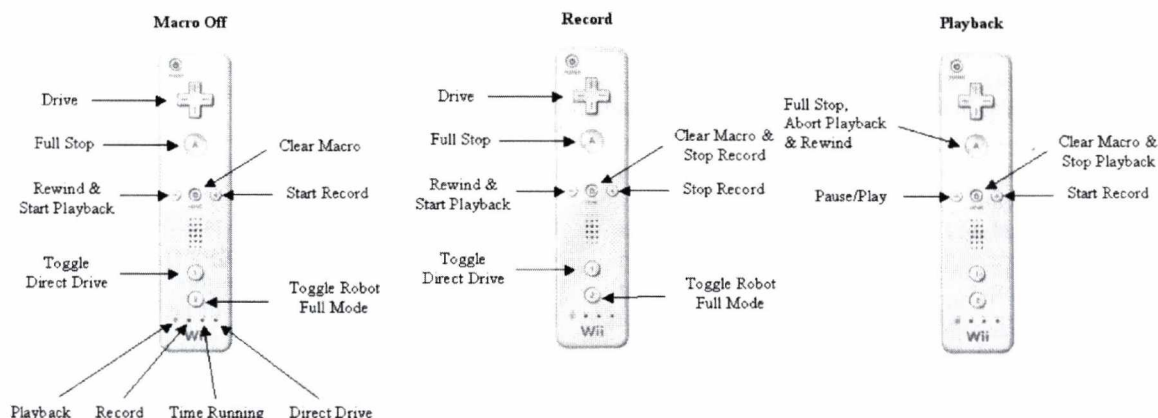


Figure 8: Using a Nintendo Wii controller and Bluetooth connection to control the motion and create programmed action sequences of the robotic camera.

- PanoCAST Inc., <http://www.PanoCAST.com> 2009. 1, 3, 5
- Takács, B., T. Fromherz, S. Tice and D. Metaxas, Digital Clones and Virtual Celebrities: Facial Tracking, Gesture Recognition and Animation for the Movie Industry, *ICCV'99*, 1999. Corfu, Greece. 6
- Takács B., "Special Education and Rehabilitation: Teaching and Healing with Interactive Graphics", *IEEE Computer Graphics and Applications*, **25**(5): pp.40-48, 2005. 5
- Takács B., Cognitive, Mental and Physical Rehabilitation Using a Configurable Virtual Reality System, *Intl. Journal of Virtual Reality*, **5**(4): pp. 1-12. 5
- Takács, B. "PanoMOBI: A Panoramic Mobile Entertainment System", *ICEC07*, Sept 15-17, 2007. Shanghai, China. 1
- Takács B. "How and Why Affordable Virtual Reality Shapes the Future of Education", *The International Journal of Virtual Reality*, **7**(1):53-66, 2008. 5
- Virtual Museum in Volos, Greece <http://www.digitalelite.us.com/Pages/PanoCAST/VirtualMuseum.html> 7
- Wikipedia Franz Kafka: The Metamorphosis, http://en.wikipedia.org/wiki/The_Metamorphosis 2009. 7

Matching Image Details with the Self Affine Feature Transform

Prohaszka Zoltan

BME IIT,
Budapest, Hungary

Abstract

Based on our research, the Self Affine Feature Transform (SAFT) was introduced as it extracts quantities which hold information of the edges in the investigated image region. This paper introduces how SAFT can be used to match interest regions of photographs. Preliminary test results are presented, which show that the performance of SAFT is close to the Scale Invariant Feature Transform (SIFT), while being computationally faster and mathematically more attractive. We highlight how SAFT can be tuned/modified for better performance.

Categories and Subject Descriptors (according to ACM CCS):
I.4.7 [Image Processing and Computer Vision]: Feature Measurement

1. Introduction

This article investigates the operation of the Self Affine Feature Transform (SAFT) on photographs, it focuses rather on matching details of photographs than analysing how geometric information is extracted from drawings. SAFT was introduced in [13].

SAFT can be effectively used to match similar details in a

transformation invariant way. This property is the main contribution of this paper. The comparison of SAFT as a feature matcher with the widely used detectors as SIFT [7], MSER [9], GLOH [12] and SURF [2] will be shown in a longer article. This paper contains only a very brief comparison to SIFT.

The introduced methods are robust, although the sensitivity against windowing phenomena cannot be neglected.

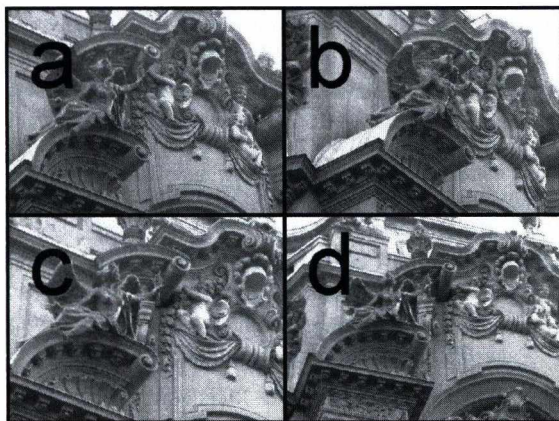


Figure 1: Typical input images

1.1. Sections Overview

Section 2 describes related work in this field. Section 3 gives the theoretical background of the SAFT detector together with formulations how to calculate it and transform it between coordinate frames. Section 4 introduces how SAFT can be used for matching details on photographs. Section 5 compares the Scale Invariant Feature Transform (SIFT) and SAFT from theoretical and experimental point of view. Section 6 shows how the introduced methods can be extended to color images.

1.2. Nomenclature

To distinguish from scalar variables, bold lower-case letters are used for vectors (\mathbf{v}) and bold capitals for matrices (\mathbf{M}). Vectors are column vectors by default, row vectors appear as

transposed columns. Kronecker product is notated by \otimes . Homogeneous quantities are denoted by subscript H , which is very useful during the implementation of algorithms working both with homogeneous and raw coordinates. C_α and S_α will refer $\cos(\alpha)$ and $\sin(\alpha)$ respectively, where α is an arbitrary rotation or angle.

Let us consider the function $\mathbf{f}(\mathbf{x})$ over the domain S , $\mathbf{x} \in S$ (both \mathbf{f} and \mathbf{x} can be column vectors or scalars). The term \mathbf{I} is the homogeneous 2nd range-, domain- or range-domain-moment of \mathbf{f} is defined by:

$$\mathbf{I} = \int \begin{bmatrix} \mathbf{v}(\mathbf{x}) \\ 1 \end{bmatrix} [\mathbf{v}(\mathbf{x})^T \ 1] \otimes g(\mathbf{x}) dS,$$

where $\{\mathbf{v}(\mathbf{x}); g(\mathbf{x})\}$ can be one of the following: $\{\mathbf{f}(\mathbf{x}); 1\}$, $\{\mathbf{x}; \mathbf{f}(\mathbf{x})\}$ or $\{\mathbf{f}(\mathbf{x}) \ \mathbf{x}; 1\}$ respectively.

Notice that the Kronecker product (\otimes) usually simplifies to multiplication by scalar, except for domain-moments if \mathbf{f} is vector valued. Range-moments represent function value histograms, while domain-moments represent the distribution of functions on the input space.

The image $I(x,y)$ is integrated according to the weighting function $w_A(x,y)$ on the domain A . Image derivatives are in vector $\mathbf{g}(x,y)$. The $:$ (colon) operator appearing in matrix subscripts denotes MATLAB[®], OCTAVE[®] style multiple indexing.

There is a quick reference in the appendix about the frequently used variables.

1.3. Positioning SAFT in the Image Processing Era

The Self Affine Feature Transform shows an interesting view when compared to other IP methods. It can be positioned among Transformation Invariant Features, since it has many similar properties, thus it can be used to solve similar tasks. However, SAFT can be used to extract exact geometric information too, thus it can be positioned for example among classic corner detectors, etc. These abilities of SAFT are due to the fact, that the calculation of the feature is based on geometrical definitions and is done by the tools of calculus and linear algebra. In certain situations, it can also be used to substitute the Hough transform. Due to the above written, SAFT cannot be positioned unambiguously among existing image processing algorithms, it has a lot of application areas.

2. Related Work

Many existing techniques (for example Edge detectors, the standard and generalised Hough transform [19], interest point detectors [10], [5], [17], and feature descriptors) can solve only a subset of the image processing tasks that SAFT can solve, and vice versa. Many similarities can be found in the formulations how the usual methods and SAFT works. Each of these similarities gives us an other viewing point of the SAFT matrix \mathbf{M} , and the information enclosed in it, see (8).

2.1. Relation to Classic Detectors

SAFT is related to classic corner detectors. The 2×2 symmetric matrix $\mathbf{C} = \mathbf{g}\mathbf{g}^T$ (\mathbf{g} is the image gradient) used by the Harris [5] and by the Shi and Tomasi [17] corner detectors also appears in the 6×6 SAFT matrix \mathbf{M} , see (8). This will be explained later in Section 3.4.1

There is also a similarity in the formulation of SAFT and affine shape adaptation [1], [11]. SAFT contains more information and changes during affine transformations in a more complex way. The philosophy of affine shape adaptation can be combined with SAFT, but one has to take into account that the computational requirement of SAFT is higher than that of simpler methods, thus considerable less iterative steps can be afforded.

2.2. Transformation Invariant Features and SAFT

Transformation invariant feature descriptors are also related to SAFT. We have to highlight SIFT [7], MSER [9], GLOH [12] and SURF [2]. They are used to match details of photographs via the comparison of the distance of feature vectors. This paper contains a very brief comparison to SIFT in Section 5. The SIFT descriptor builds gradient histograms at characteristic positions of the investigated detail. The resulting 128 dimensional parameter vector is compared elementwise during matching. Speed up Robust Features (SURF) was inspired by SIFT, it uses many simplifications. The resulting parameter vector's length is 64, but the authors tested many options regarding parameter vector size, these are referred as SURF-36, SURF-128, for example. Gradient Location and Orientation Histogram (GLOH) builds statistics of image derivatives over a polar grid. gradients are collected into 16 orientation bins at 17 different locations. The resulting 272 dimension descriptor was analysed with PCA over 47000 images, and the most important 128 dimension was kept in the final descriptor. Maximally Stable Extremal Regions does not descends from SIFT. It operates based on the intensity image and finds regions which are stable against continous geometric and monotonic intensity transformations.

2.2.1. Invariant Moments

SAFT is also related to Affine Moment Invariants [3], [16] and to Color Image Moments [18]. The highest order moment utilized in SAFT is the 2nd, while the others use higher degree moments also. Affine Moment Invariants use only moments calculated over the intensity image, while Color Image Moments over the multichannel image:

$$\mathbf{M}_{ij}^{klm} = \int x^i y^j R^k G^l B^m dA,$$

where $R(x,y)$ is the red channel of the image, for example. In contrast, SAFT operates on the gradient images, see (10).

2.3. SAFT and the Lucas-Kanade Detector

Relation of SAFT to the Lucas-Kanade (LK) detector is essential, since both the formulation of SAFT and its basic idea descends from the affine LK detector. Already the first paper of Lucas and Kanade [8] suggest the affine extension of their method, to determine the infinitesimal affine transformation between two similar portion of images.

3. Basic Relations of the SAFT Detector

SAFT was introduced by describing the invariance of an image to different affine flows [13]. This invariance can be computed by the affine LK detector.

3.1. Affine Flows: Definition

The basic formulation of the SAFT feature comes from an image's invariance against infinitesimal affine transformations. *Infinitesimal affine transformations will be referred as affine flows, which satisfy that the local velocity depends linearly both on homogeneous position and on flow parameters:*

$$\mathbf{v} = \mathbf{Q} \cdot \mathbf{p}_H = [\mathbf{p}_H^T \otimes \mathbf{I}_{2 \times 2}] \cdot \mathbf{q} \quad (1)$$

where \mathbf{v} is 2×1 column vector of local velocity, \mathbf{Q} is 6 DoF 2×3 parameter matrix, $\mathbf{p}_H = [x \ y \ 1]^T$ is 3×1 homogeneous position and \mathbf{q} contains the elements of \mathbf{Q} in column-major order, thus \mathbf{q} is the parameter vector. It is possible to give an alternative definition:

$$\mathbf{v} = [\mathbf{I}_{2 \times 2} \otimes \mathbf{p}_H^T] \cdot \hat{\mathbf{q}}, \quad (2)$$

with reversed order of Kronecker product, where $\hat{\mathbf{q}}$ contains the element of \mathbf{Q} in row-major order. All matrices descending from this alternative definition will be notated by hats ($\hat{\cdot}$). \mathbf{Q} will be decomposed as

$$\mathbf{Q} = [\mathbf{F} \ \mathbf{t}] = \begin{bmatrix} s_x & r_x & t_x \\ r_y & s_y & t_y \end{bmatrix}, \quad \mathbf{q} = \begin{bmatrix} \mathbf{f} \\ \mathbf{t} \end{bmatrix}, \quad \mathbf{Q}_H = \begin{bmatrix} \mathbf{Q} & 0 \\ 0 & 0 & 0 \end{bmatrix}. \quad (3)$$

The elements of \mathbf{F} is collected to \mathbf{f} in column-major order ($\mathbf{f} = [s_x \ r_y \ r_x \ s_y]^T$).

3.1.1. Coordinate Transformations on Affine Flows

We can describe the same flow in coordinate frames A and B , which have the relation:

$$\mathbf{p}_{HB} = \mathbf{T}_{HBA} \mathbf{p}_{HA}, \quad \mathbf{T}_{HBA} = \begin{bmatrix} \mathbf{R}_{BA} & -\mathbf{c}_{BA} \\ 0 & 0 & 1 \end{bmatrix} \quad (4)$$

where \mathbf{R}_{BA} is not necessarily orthogonal. The transformation rules for \mathbf{Q} and \mathbf{q} , $\hat{\mathbf{q}}$ are

$$\mathbf{Q}_B = \mathbf{R}_{BA} \mathbf{Q}_A \mathbf{T}_{HBA}^{-1},$$

$$\mathbf{q}_B = \mathbf{S}_{BA} \mathbf{q}_A, \quad \mathbf{S}_{BA} = \mathbf{T}_{HBA}^{-T} \otimes \mathbf{R}_{BA} \quad (5)$$

$$\mathbf{q}_A = \mathbf{S}_{AB} \mathbf{q}_B, \quad \mathbf{S}_{AB} = \mathbf{S}_{BA}^{-1}$$

$$\hat{\mathbf{S}}_{BA} = \mathbf{R}_{BA} \otimes \mathbf{T}_{HBA}^{-T}, \quad \hat{\mathbf{q}}_B = \hat{\mathbf{S}}_{BA} \hat{\mathbf{q}}_A \quad (6)$$

The transformation \mathbf{S} (and $\hat{\mathbf{S}}$) are unitary, if and only if the affine transformation encodes rotation and scaling around the origin.

3.2. Algebraic Background of the SAFT Descriptor

The formulation of SAFT originates from the affine extension of the LK detector, thus we describe the essential equations of the LK detector (with the notations of this paper) and then give the definition of SAFT.

3.2.1. Formulation of the Lucas-Kanade Detector

The generalised LK detector can determine the linear parameters of the optical flow between two images. The gradients and difference of these images are used in the calculations. The detector integrates quadratic error functions of flow parameters arising from the squared equation error of linear constraints against local flow velocities. Thus, the affine extension determines a quadratic cost function on the parameter vector \mathbf{q} . The primary output of any LK detector is the homogeneous quadratic form of the reprojection error's dependence from flow parameters.

Usually, the extensions of the LK detector assume that $\mathbf{v}(\mathbf{p})$, the local flow velocity depends on planar position \mathbf{p} and linearly on flow parameters \mathbf{q} .

$$\mathbf{v}(\mathbf{p}) = \mathbf{L}(\mathbf{p}) \mathbf{q} \quad (7)$$

In the standard detector $\mathbf{L}_{STD}(\mathbf{p}) = \mathbf{I}_{2 \times 2}$, $\mathbf{q}_{STD} = [t_x \ t_y]^T$. The affine extension can use for example $\mathbf{L} = \mathbf{p}_H^T \otimes \mathbf{I}_{2 \times 2}$ and $\mathbf{q} = [s_x \ r_y \ r_x \ s_y \ t_x \ t_y]^T$. The LK detector integrates squared reprojection error at image points:

$$e^2 = \int (\mathbf{v}(\mathbf{p})^T \mathbf{g}(\mathbf{p}) - \Delta I(\mathbf{p}))^2 w(\mathbf{p}) dA$$

where $\Delta I(\mathbf{p})$ is the difference between the two images at point \mathbf{p} , $\mathbf{g} = \nabla I(\mathbf{p})$ is the (averaged) gradient of the image(s), and $w(\mathbf{p})$ is the applied windowing function. This results

$$e^2 = [\mathbf{q}^T \ 1] \begin{bmatrix} \mathbf{M} & \mathbf{n} \\ \mathbf{n}^T & h \end{bmatrix} \begin{bmatrix} \mathbf{q} \\ 1 \end{bmatrix},$$

$$\begin{bmatrix} \mathbf{M} & \mathbf{n} \\ \mathbf{n}^T & h \end{bmatrix} = \int \begin{bmatrix} \mathbf{L}(\mathbf{p})^T \mathbf{g}(\mathbf{p}) \\ -\Delta I(\mathbf{p}) \end{bmatrix} \begin{bmatrix} \mathbf{g}(\mathbf{p})^T \mathbf{L}(\mathbf{p}) & -\Delta I(\mathbf{p}) \end{bmatrix} w dA.$$

Minimising this error leads the relation $\mathbf{q}_{opt} = \text{argmin}(e^2(\mathbf{q}))$, $\mathbf{q}_{opt} = -\mathbf{M}^{-1} \mathbf{n}$, but the full information extracted from the images is carried by \mathbf{M} and \mathbf{n} . Usually only the solution of the above equation is returned by the LK implementations. As described above, we are interested in the linear constraints against this solution, so we will use the full information encapsulated in matrix \mathbf{M} .

3.2.2. Calculating the SAFT Descriptor

The LK detector assumes that two, slightly different image will be analyzed to determine the optimal optical flow between them. However, all equations and assumptions remain valid if we feed the same picture to the detector instead of two different images. Why would one do so, as it is obvious that the optimal solution is the zero flow, as $\mathbf{n} = \mathbf{0} \Rightarrow \mathbf{q}_{opt} = \mathbf{0}$ in this case? The reason is, that we want to investigate, how does the reprojection error depend if we disturb flow parameters around the optimum $\mathbf{q}_{opt} = \mathbf{0}$.

$$\mathbf{M} = \int (\mathbf{p}_H \otimes \mathbf{g})(\mathbf{p}_H \otimes \mathbf{g})^T w dA \quad (8)$$

$$\hat{\mathbf{M}} = \int (\mathbf{g} \otimes \mathbf{p}_H)(\mathbf{g} \otimes \mathbf{p}_H)^T w dA$$

The matrix \mathbf{M} (and $\hat{\mathbf{M}}$) will be block symmetric with 18 independent elements and positive semi-definite. The squared total error can be expressed as:

$$e^2 = \int e(\mathbf{p})^2 w dA = \mathbf{q}^T \mathbf{M} \mathbf{q} = \hat{\mathbf{q}}^T \hat{\mathbf{M}} \hat{\mathbf{q}}.$$

Prior to measuring gradients a pre-processing convolution filter is used. A Gaussian bell is applied with parameter σ_{dif} .

3.2.3. Coordinate Transformations

The behavior of the algorithm depends on the chosen coordinate frame. The dependence is the following:

$$\mathbf{M}_B = \mathbf{S}_{AB}^T \mathbf{M}_A \mathbf{S}_{AB}, \quad \hat{\mathbf{M}}_B = \hat{\mathbf{S}}_{AB}^T \hat{\mathbf{M}}_A \hat{\mathbf{S}}_{AB} \quad (9)$$

where \mathbf{S}_{AB} and $\hat{\mathbf{S}}_{AB}$ are the same as in (5) and (6).

3.3. Normalising Flow Strength

As flows with parallel vectors of different flow strength ($\mathbf{q}_2 = c\mathbf{q}_1$, $c \in \mathbb{R}$) result different self-affine errors ($e_2^2 = c^2 e_1^2$) proportional to c^2 , flow strength should be normalised. To do so flow strength measure has to be defined. The length of parameter vector \mathbf{q} is proportional to average squared flow velocity strength in the investigation window if the window is:

- disk with $r_{max} = 2$
- Gaussian bell with $\sigma = \sqrt{2}$, $r_{max} \gg 1$
- square with sides $2\sqrt{3}$
- Hann(ing) window $w = (1 + \cos(\pi\sqrt{x^2 + y^2}/r_{max}))/2$ with $r_{max} \approx 2 \cdot 1.4655$

Choosing the window and the coordinate frame according to the above ensures that any two flow parameter vectors with the same length represent the same average flow strength.

Remark I: This condition is assumed in the rest of this article.

The results above determine the optimal selection of the coordinate frame versus the investigation window. Therefore, the coordinate unit is defined relative to the size of the window.

Remark II: In the following, the term *unit (length)* will refer the coordinate unit resulting from the deductions above.

3.4. Alternative Interpretations

This subsection gives alternative definitions of the introduced entities and quantities. \mathbf{M} also can be interpreted as the homogeneous 2nd domain-moment of $\mathbf{g}\mathbf{g}^T$.

$$\hat{\mathbf{M}} = \int [\mathbf{g}\mathbf{g}^T] \otimes [\mathbf{p}_H \mathbf{p}_H^T] \cdot w dA, \quad (10)$$

Similarly $\mathbf{M} = \int [\mathbf{p}_H \mathbf{p}_H^T] \otimes [\mathbf{g}\mathbf{g}^T] \cdot w dA$. The block symmetric property is clearly seen on the above formulae. Although $\hat{\mathbf{M}}$ (and \mathbf{M}) is the sum of Kronecker products, the spectral decomposition theorem of Kronecker products [6] cannot be utilized, since the required characteristics is lost during integration. Only the block symmetric property is preserved.

A 9×9 extension of \mathbf{M} can be defined:

$$\mathbf{M}^* = \int (\mathbf{p}_H \cdot \mathbf{p}_H^T) \otimes (\mathbf{g}_H \cdot \mathbf{g}_H^T) w dA$$

where $\mathbf{g}_H = [g_x \ g_y \ I]^T$ and I is the image intensity. \mathbf{g} can also be extended as $\mathbf{g}_H = [g_x \ g_y \ 1]^T$. In this case the lower right symmetric 3×3 block of \mathbf{M}^* describes only the shape of the interest window itself, not its contents.

3.4.1. Decomposition of $\hat{\mathbf{M}}$ and \mathbf{M}

The 18 independent elements of $\hat{\mathbf{M}}$ can be compressed by the decomposition:

$$\hat{\mathbf{M}} = \begin{bmatrix} \hat{\mathbf{M}}_{xx} & \hat{\mathbf{M}}_{xy} \\ \hat{\mathbf{M}}_{xy} & \hat{\mathbf{M}}_{yy} \end{bmatrix} = \frac{1}{2} \begin{bmatrix} \mathbf{G}_K + \mathbf{G}_C & \mathbf{G}_S \\ \mathbf{G}_S & \mathbf{G}_K - \mathbf{G}_C \end{bmatrix} \quad (11)$$

where \mathbf{G}_K , \mathbf{G}_C and \mathbf{G}_S are symmetric, and \mathbf{G}_K is the homogeneous domain-moment of the squared gradient magnitude.

$$\mathbf{G}_K = (\hat{\mathbf{M}}_{xx} + \hat{\mathbf{M}}_{yy}), \quad \mathbf{G}_C = (\hat{\mathbf{M}}_{xx} - \hat{\mathbf{M}}_{yy}), \quad \mathbf{G}_S = (\hat{\mathbf{M}}_{xy} + \hat{\mathbf{M}}_{xy}).$$

Each of this 3×3 symmetric matrices can be decomposed similarly:

$$\mathbf{G}_i = \frac{1}{2} \begin{bmatrix} g_{iK} + g_{iC} & g_{iS} & g_{ix} \\ g_{iS} & g_{iK} - g_{iC} & g_{iy} \\ g_{ix} & g_{iy} & g_{i1} \end{bmatrix}$$

where subscript i can be either K , C or S . These elements can be arranged to:

$$\mathbf{m}_i = [g_{iK} \ g_{iC} \ g_{iS} \ g_{ix} \ g_{iy} \ g_{i1}]^T.$$

Hence we obtain $6 \times 3 = 18$ independent scalars to describe $\hat{\mathbf{M}}$. Vectors \mathbf{m}_K , \mathbf{m}_C and \mathbf{m}_S can be enclosed to:

$$\mathbf{G} = [\mathbf{m}_K \ \mathbf{m}_C \ \mathbf{m}_S] \quad (12)$$

The 6×3 matrix \mathbf{G} describes all information enclosed to $\hat{\mathbf{M}}$ (and \mathbf{M}).

Let \mathbf{C} , the 2×2 symmetric matrix denote the lower-right part of \mathbf{M} :

$$\mathbf{C} = \mathbf{M}_{5\dots6,5\dots6} \quad (13)$$

\mathbf{C} is frequently used in keypoint detectors, for example in the Harris [5] and in the Shi and Tomasi [17] corner detectors. It is also the fundamental output of the standard LK problem, as $e^2 = [t_x \ t_y] \mathbf{C} [t_x \ t_y]^T$ describes how sensitive is the image detail against different translations. The sum of squared errors of two uniform shifts in any two perpendicular directions can get by:

$$E_{AC} = \text{trace}(\mathbf{C}) = e_{t_x}^2 + e_{t_y}^2, \quad (14)$$

which is the accumulated energy of the 'AC' component in the image.

\mathbf{C} can also be interpreted as the 2nd order component of the gradient histogram (range-moment) of the image ($\int \mathbf{g} \mathbf{g}^T \text{wd}A$), and hence:

$$E_{AC} = g_{K1}/2$$

3.5. Similarity transformations on \mathbf{G}

If the SAFT feature is evaluated in the same region, but in a different coordinate frame, which is scaled and rotated respect to the original one, then $\hat{\mathbf{S}}_{BA}$, the matrix describing the relation between $\hat{\mathbf{M}}_A$ and $\hat{\mathbf{M}}_B$ will be orthogonal. The elements in \mathbf{G} change with coordinate frame scaling ($c \in \mathbb{R}$) and rotation (α) according to the following:

$$\mathbf{G}_B = \begin{bmatrix} \mathbf{Z}_3(\alpha) & \mathbf{0}_{3 \times 3} \\ \mathbf{0}_{3 \times 3} & c^2 \mathbf{T}_{BA}^T \end{bmatrix} \mathbf{G}_A \mathbf{Z}_3(\alpha), \quad \mathbf{T}_{BA} = \begin{bmatrix} c \mathbf{R}_\alpha & \mathbf{0} \\ \mathbf{0}^T & 1 \end{bmatrix} \quad (15)$$

$$\mathbf{R}_\alpha = \begin{bmatrix} C_\alpha & -S_\alpha \\ S_\alpha & C_\alpha \end{bmatrix}, \quad \mathbf{Z}_3(\alpha) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & C_{2\alpha} & S_{2\alpha} \\ 0 & -S_{2\alpha} & C_{2\alpha} \end{bmatrix}$$

This simple transformation rule is the reason, why we chose representation \mathbf{G} according to (12). Several rotation invariant quantities can be found, for example g_{KK} , $g_{CK}^2 + g_{SK}^2$, etc.

4. Matching features with SAFT

The 18 independent element of \mathbf{G} was not found to be enough for matching image details based on experiments. These experiments localized interest windows by Lowe's DoG method [7]. Thus, the location and size of the windows were fixed during the tests. However, the 'wavelength' (σ_{dif}) of the pre-processing Gaussian filter was varied during the tests. These variation led us to the idea, to evaluate \mathbf{G} , the SAFT feature at different spatial frequencies.

We found 3 filters, which produced a SAFT feature-triplet, that contained enough information required for reliable, robust matching. This gives a feature vector length of

round	1 st	2 nd	3 rd
r_w , window radius	20	17	10
σ_{dif}	0.25	1	2.8
σ_{int}	$r_w/3$	$r_w/3$	$r_w/3$

Table 1: SAFT filter settings for resampling, differentiating and integration for different rounds. All quantities are given in resampled pixels.

54 elements, which is considerably smaller than the 128 elements of SIFT. Table 1 shows settings for each member of the feature triplet. These setting was found to perform the best among those we tested, and preformed comparable to SIFT. The test implementation used a preliminary scaling of the diagonal of $\hat{\mathbf{M}}$ by 1.5 before calculating \mathbf{G} . Optimal scaling of the elements in \mathbf{G} was described in [14], which scaling was not tested yet. Each 18-dimensional feature vector has been normalised in the test setup. Other possibilities are normalising the 54-dimensional feature vector, or dividing each element by $E_{AC_{2nd}}$ or by $\sum_{i=1}^3 E_{AC_{ph}}$. It has to be taken into account, that larger σ_{dif} values are resulting lower cut-off frequencies, thus, the gradient magnitudes will be smaller. One might apply the transformations:

$$\mathbf{G}'_{1st} = \mathbf{G}_{1st}, \quad \mathbf{G}'_{2nd} = \mathbf{G}_{2nd} - \mathbf{G}_{1st}, \quad \mathbf{G}'_{3rd} = \mathbf{G}_{3rd} - \mathbf{G}_{2nd},$$

The new descriptors represent different frequency bands, which have gradient strength proportional to their medium frequency. The 18 Dof descriptors therefore needs to be scaled according to this effect, before combining the 54-dimensional descriptor.

Remark: $\sigma_{int} = r_w/3$ does not contradict to $\sigma_{int} = \sqrt{2} \text{units}$, rather it defines the ratio of the optimal coordinate unit to r_w .

4.1. Optimal Rotation

One benefit of using analytical moments as feature descriptors over histograms that several post processing steps can be applied after feature evaluation. One such step is rotating the feature descriptor according to interest window rotation. Section 3.5 describes the algebraic relations in this case. This setup has the advantage, that the rotation between two features can be varied during matching, and an optimal rotation angle can be found based on best matching candidate. Unfortunately, closed expression for optimal rotation can not be found for general cases, however, the computational requirement of the exhaustive search for best angle alignment can be decreased considerably.

Furthermore, during guided matching (this step is performed if a preliminary registration has been acquired, and the current task is to find best pair based on preliminary hypothesis, [4]) the relative rotation between two matching

candidates can be obtained from the preliminary transformation, and features can be considered pairs only if they descriptor align when evaluated with the relative angle coming from the preliminary transformation.

Determining optimal rotation was not tested during feature matching yet, we used the angles calculated by SIFT during our first experiments. Our next paper will contain the comparison of the two methods for determining the relative angle of two similar feature.

5. Comparison with SIFT

This paper focuses on the SIFT and SAFT feature descriptor calculation and performance. Interest window localization was done in our test in the same way for each descriptor. We used the codes of Andrea Vedaldi [20] for localizing interest windows and extracting SIFT features. SAFT features was extracted by our code. Run times will not be compared, while the original code use C functions called from MATLAB[®], while our code is running on the interpreter.

In the following, we distinguish between the two main steps of SIFT by using the expressions SIFT keypoint localisation and SIFT descriptor calculation.

5.1. Computational requirement

With the current parametrization of SAFT, calculating one SAFT descriptor takes more operations than calculating one SIFT descriptor. However, due to the shorter parameter vector, SAFT becomes faster if the computational requirement of exhaustive matching is also considered. Suppose, that for n images one would like to match each image with m other images, while having d descriptors in each image.

Feature extraction time is proportional to $n \cdot d$ while matching time is proportional to $n \cdot m \cdot d^2/2$. If $m = 3$ and $d = 2000$, the advantage of SAFT to SIFT is 888 million mul/add operations per image during matching, while the disadvantage during feature extraction is approximately 150 million simple operations per image.

Remark I.: SAFT does not use transcendent functions during feature extraction, while SIFT uses $\text{atan2}()$ and $\text{sqrt}()$. **Remark II.:** The computational requirement of SAFT during feature extraction might be lowered by tuning certain parameters.

5.2. Invariance

The affine invariance of SIFT descriptor is limited, but is reported to be robust enough for matching rotated 3D objects [7]. This invariance is due to the gradient histogram bins, which give very similar results for slightly distorted image content. The affine invariance of SAFT comes from the analytical evaluation. In [13] and [15] it is highlighted, that

the SAFT matrix encodes image information in an affine invariant manner. Any SAFT feature can be post processed as it were evaluated in a different coordinate frame (9). Thus, the descriptor itself is analytically invariant against translation, rotation, scaling, affinity and shear. However, this invariance of SAFT is weakened due to the used differentiation and integration kernels and the used SIFT feature localisation code. Thus, the whole algorithm is absolutely robust against rotation, scaling and translation, and slightly invariant against affinity as in the case of SIFT.

We observed the effects of all phenomenon described above during our tests. We did not find any of the two feature considerably superior in invariance against affinity and spatial rotation of objects when they was used to match photographs. However, the formulation of SAFT has undiscovered possibilities to extend the algorithm to be more robust against the mentioned effects. For example, during guided matching, (9) can be applied to compare features in coordinate frames according to the preliminary, locally affine geometric transformation. This method could be extended to modify the integration window also.

5.3. Extracting geometric information

SAFT is able to extract various geometric information from image details, if the detail consists of a few colored areas bounded by analytical curves. This ability of SAFT is detailed in [13] and [15]. It can find the followings by closed expressions: equation of conic sections, vanishing point of convergent lines, circles which fit best to image curves, etc. As C can be reproduced from the SAFT feature, it can be used, to determine the uncertainty (or covariance) of the observed position of the given feature. As the invariance of the image detail against affine transformations is incorporated in the SAFT descriptor, this information can be utilised when calculating the optimal piecewise affine dense matching between two images resulting minimal SSD pixel error.

The SIFT descriptor has no similar abilities.

5.4. Test Results

This paper describes only the first test results of the SIFT-SAFT comparison. A directory processing algorithm was used to compare successive image pairs of a photograph sequence and find the epipolar geometry which describes the relation between them. Feature localisation was done by Lowe's DoG method. Feature orientation calculation was also done by the algorithm of Lowe to enable the pure comparison of descriptor performance. Feature extraction was done either by SIFT or by SAFT with the parameters described in Section 4. Feature matching was done by elementwise comparison. Only matches with relatively far second closest match were considered, according to Lowe's recommendation. Distance threshold was set to 1.5.

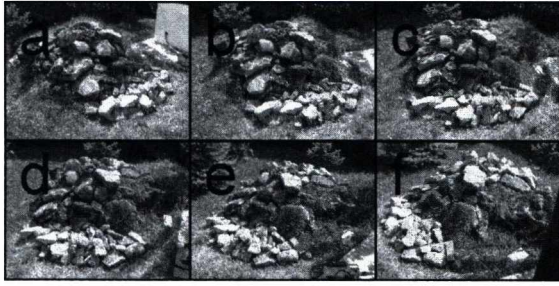


Figure 2: Second image sequence used in the tests of Table 2.

The quantities listed in Table 2 show that the two features behave comparable in 5 cases out of 9, slightly worse in cases 1d-1a and 2a-2b, while SIFT finds more matches in the remaining 2 cases: 1b-1c and 2e-2f. When only the largest 250 keypoints are matched, SAFT gets more closer in performance. SAFT performance depends on the size of the keypoints, because the 1st round of SAFT feature uses finer resampling than SIFT, resulting that for small keypoints, the 1st round does not contain useful information. Image sequences were not selected based on SAFT performance, but were chosen independently.

5.5. Adjustable parameters

The following parameters affect the performance of SAFT as photograph matcher:

- The ratio of σ_{int} vs. the size of the localized feature. The code of Vedaldi uses SIFT descriptor area radius which is 6 times bigger than the localized size. Our application uses the same window, which results that σ_{int} is the double of the localized size.
- Two parameters appearing in Table 1: r_w and σ_{dif} , for each round.
- Decreasing the ratio of r_w to σ_{int} decreases computational complexity, but increases windowing effects.
- The scaling of the elements of \mathbf{G} . (Changing the ratio of the coordinate unit and σ_{int} can be interpreted as one kind of scaling.)
- The normalisation of the 54-dimensional feature vector. See Section 4.
- The minimum required ratio of the closest distance to the second closest distance.

Up to this date, only the effect of r_w and σ_{dif} was investigated in details, while other parameters were not varied. Detailed tests are to be carried out in the near future.

6. Color Channels

Until this point, we supposed that the image consists of one (grayscale) channel. As for the LK detector, the simple addition of the resultant \mathbf{M} matrix of every channel results the

correct propagation of sum of squared pixel errors, measured channelwise.

$$\mathbf{M}_{RGB} = \mathbf{M}_R + \mathbf{M}_G + \mathbf{M}_B$$

The resultant SAFT matrix \mathbf{M}_{RGB} encodes more information than \mathbf{M}_{gray} of the grayscale version of the multichannel image. Generally, the above formulation fits to geometric information extraction from SAFT. When SAFT is used as feature detector, one might want to compare the feature vector (or matrix) of each color channel separately. The tests described in this article used \mathbf{M}_{gray} , because otherwise it could not be compared to the performance of SIFT. The performance of SAFT based on \mathbf{M}_{RGB} and channelwise feature comparison needs to be tested.

7. Implementation

The authors plan to publish the developed MATLAB[®] (and C++) compatible algorithms in the near future.

8. Conclusions

This paper has described the basic extension of the SAFT descriptor which enables using it to match photographs. It reports our first comparison to SIFT. Not all parameters were optimized, the parameters used in the presented tests are the first parameter set which performed close to SIFT while being computationally simpler.

A detailed article of this subject is expected to be contributed in the near future, which will contain more test results about the tuning of SAFT parameters together with detailed comparisons to other feature descriptors.

Acknowledgments

The research in this article was supported by the Hungarian National Research Program grant No. OTKA K 71762.

References

- [1] A. Baumberg, *Reliable feature matching across widely separated views*, IEEE Conference on Computer Vision and Pattern Recognition, 2000, pp. 1774–1781. ↑2
- [2] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, *SURF: Speeded Up Robust Features*, Computer Vision and Image Understanding (CVIU) **110** (2008), no. 3, 346–359. ↑1, 2
- [3] J. Flusser and T. Suk, *Pattern recognition by affine moment invariants*, Pattern recognition **26** (1993), no. 1, 167–174. ↑2
- [4] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, Second, Cambridge University Press, Cambridge, UK., 2004. ↑5
- [5] C. Harris and M. Stephens, *A combined corner and edge detector*, 4th Alvey Vision Conference, 1988, pp. 147–151. ↑2, 5
- [6] A. J. Laub, *Matrix analysis for scientists and engineers*, Society for Industrial and Applied Mathematics, Philadelphia, PA, 2005. ↑4

image pairs	localized features	matched*	SIFT inliers*	inlier error	matched*	SAFT inliers*	inlier error
1a-1b	1878-1908	525[72]	437[60]	0.526437	628[87]	387[44]	0.506873
1b-1c	1908-2052	332[40]	226[25]	0.705524	433[53]	70[21]	0.693230
1c-1d	2052-1975	650[78]	560[61]	0.426955	689[92]	458[58]	0.571921
1d-1a	1975-1878	409[52]	316[30]	0.477635	486[61]	183[28]	0.672835
2a-2b	2049-2099	242[54]	192[42]	0.553002	374[73]	113[31]	0.613243
2b-2c	2099-2135	269[79]	222[58]	0.575525	443[87]	206[44]	0.626815
2c-2d	2135-2111	263[59]	225[44]	0.501209	396[72]	181[42]	0.682369
2d-2e	2111-2174	197[60]	166[43]	0.608471	348[70]	120[20]	0.759219
2e-2f	2174-2141	264[47]	216[30]	0.710936	416[70]	93[14]	0.829069

Table 2: Test results for sequences seen in Figure 1 and Figure 2. Inlier error: squared mean inlier distance from model, relative to the inlier limit, which was set to 1/400 of the image diagonal. *: Elements in brackets are got by matching only the 250 largest localized feature.

[7] D. Lowe, *Distinctive image features from scale-invariant keypoints*, International Journal of Computer Vision **60** (2004), no. 2, 91–110. ↑1, 2, 5, 6

[8] B. D. Lucas and T. Kanade, *An iterative image registration technique with an application to stereo vision*, Imaging Understanding Workshop, 1981, pp. 121–130. ↑3

[9] J. Matas, O. Chum, M. Urban, and T. Pajdla, *Robust wide-baseline stereo from maximally stable extremal regions*, British Machine Vision Conference (Cardiff, UK), 2002, pp. 384–393. ↑1, 2

[10] S. El Mejdani, R. Egli, and F. Dubeau, *Old and new straight-line detectors: Description and comparison*, Pattern Recogn. **41** (2008), no. 6, 1845–1866. ↑2

[11] K. Mikolajczyk and C. Schmid, *Scale & affine invariant interest point detectors*, International Journal on Computer Vision **60** (2004), no. 1, 63–86. ↑2

[12] K. Mikolajczyk and C. Schmid, *A performance evaluation of local descriptors*, IEEE Transactions on Pattern Analysis and Machine Intelligence **10** (2005), no. 27, 1615–1630. ↑1, 2

[13] Z. Prohaszka, *Affine Invariant Features from Self-Flow*, RAAD 17th International Workshop on Robotics in Alpe-Adria-Danube Region (Ancona, Italy), 2008, pp. 1–10. <http://mycite.omikk.bme.hu/doc/40788.pdf>. ↑1, 3, 6

[14] Z. Prohaszka, *Fine Tuning of Quasi Linear Feature Descriptors*, 7th Conference of Hungarian Association for Image Processing and Pattern Recognition (KEPAF2009) (Budapest, Hungary), 2009, pp. 1–8. <http://mycite.omikk.bme.hu/doc/69862.pdf>. ↑5

[15] Z. Prohaszka and B. Lantos, *Extracting geometric information from images with the novel Self Affine Feature Transform*, Submitted to: Periodica Polytechnica, Electrical Engineering (2009). ↑6

[16] E. Rahtu, M. Salo, J. Heikkil, and J. Flusser, *Generalized affine moment invariants for object recogn*, 18th International Conference on Pattern Recognition (ICPR'06), 2006, pp. 634–637. ↑2

[17] J. Shi and C. Tomasi, *Good Features to Track*, 9th IEEE Conference on Computer Vision and Pattern Recognition, 1994, pp. 593–600. ↑2, 5

[18] J-L. Shih and L-H. Chen, *Color Image Retrieval Based on Primitives of Color Moments*, Recent Advances in Visual Information Systems (2002), 19–27. ↑2

[19] T. Tuytelaars, M. Proesmans, and L. V. Gool, *The cascade Hough transform*, ICIP, 1998, pp. 736–739. ↑2

[20] A. Vedaldi, *SIFT for Matlab*. <http://www.vlfeat.org/~vedaldi/code/sift.html>. ↑6

Appendix

8.1. Frequently used variables

	size	description	Eq.
C	2 × 2	lower-right part of M	(13)
g(x,y)	2 × 1	gradient of the image at x,y	
G	6 × 3	SAFT feature descriptor	(12)
G_C, G_S	3 × 3	domain-moment of horizontal and vertical gradients	(11)
G_K	3 × 3	domain-moment of gradient strength	(11)
M, M̂	6 × 6	SAFT descriptor matrix	(8)
p_H	3 × 1	Homog. planar position in current frame [x y 1] ^T	(1)
Q	2 × 3	Matrix of affine flows	(1)
q, q̂	6 × 1	affine flow parameters, elements of Q	(1)
S, Ŝ	6 × 6	Matrices transforming q, M and q̂, M̂	(5),(6)
T_H	3 × 3	Homog. affine coordinate transformation matrix	(4)
w(x,y)	1 × 1	windowing function of the image at x,y	
Z₃(α)	3 × 3	Matrix transforming G	(15)

Motion Detection Using Low Resolution Video Sequences

T. Helfenbein and F. Vajda

Department of Control Engineering and Information Technology,
Budapest University of Technology and Economics, Budapest, Hungary

Abstract

The paper discusses technologies of motion detection focusing on image processing methods for real-time motion detection using camera images. Popular image processing methods like SAD or estimation of the optical flow have significant disadvantages in low resolution video sequence applications. The extension of the SAD method, called WSSAD is introduced in this paper. This algorithm reduces the number of false detections. The WSSAD and the SAD method is tested and compared using an experimental motion sensor module.

Categories and Subject Descriptors (according to ACM CCS): I.2.10 [Vision and Scene Understanding]: Motion

1. Introduction

Motion sensors are prominently important parts of security, home care and Ambient Assisted Living systems. These sensors monitor our state and environment 24 hours a day. In these domains, besides motion sensors operating by infrared principle, camera systems become popular.

The active infrared sensor emits infrared light and measures the reflected intensity. The passive measures the light intensity of the environment in the infrared domain. The sensitivity characteristics of these sensors are not linear in the measurement range. The manufacturers are trying to reduce this non-linearity using optical solutions. This technology also have problems with infrared emission of external light sources and air movement. Most of the infrared sensors have high false detection rate due to its sensitivity to these environmental effects. In these cases the sensor detects motion but only the lighting has changed or hot wind has blown.

Sensing visible light using a camera is the other operating principle of motion detection. Camera systems monitors the scene permanently and produces images from time to time. This video sequence is the input of motion detection algorithms, which typically use image processing to identify motion in the observed area. The goal of motion detection is not to detect or track all objects on the scene but produce a sequence of binary values (motion / no motion) from at least two sequential images.

Unfortunately, these systems also have disadvantages like sensitivity to lighting condition changes. In addition, application of high resolution cameras is not a cost effective solution, because they require fast and expensive processing hardware, such as DSP-s or FPGA-s. Motion detection using low resolution camera images and processing with a microcontroller can be suitable to reduce costs, but raises some other issues. Image processing methods like the estimation of the optical flow fails in this context due to the low resolution of the images. It needs high processing speed, which a controller can not ensure. In contrast, methods like SAD (Sum of Absolute Differences) can be executed on a simple microcontroller, but they are sensitive to lighting changes.

This paper focuses on image processing methods of real-time motion detection using video sequences or camera images.[1] We introduce problems, solutions and implementation proposals for motion detection using low resolution images.

2. Motion detection using video sequences

Effective sensors uses simple methods to measure particular quantities of the environment as fast and reliable as possible. We discuss the methods, which use sequential images as an input and produce logical output of two possible values: MOTION, NO_MOTION.

Motion detection methods can extract wide range of

image features. Sappa et al. introduced a method which uses the extraction of edges to detect motion.[2] A type of methods models background and compares it to the actual image.[3] Ulges at al. introduced a background modeling technique based probability calculation of the dominant motion.[4]

If two sequential images (I_A , I_P) are given, the most plausible solution is to subtract the intensity of each pixel of the images. The result is an image called difference image and contains the intensity differences between the sequential video frames. The Sum of Absolute Differences (SAD) method is based on this principle.[1] This sums the absolute value of the pixels of the difference image, and if this value is greater than a threshold value the result is MOTION. The SAD can be calculated using the following formula.

$$SAD = \sum_{y=1}^H \sum_{x=1}^W |I_{A,x,y} - I_{P,x,y}| \quad (1)$$

where the actual and previous images are I_A and I_P , H is the height, W is the width of these images. The result R can be calculated using threshold value T .

$$R = \begin{cases} \text{MOTION} & \text{if } SAD > T \\ \text{NO MOTION} & \text{otherwise} \end{cases} \quad (2)$$

This method needs at least one image to store to calculate the difference image. This image is typically the previous image in the image sequence. After a calculation step, the stored image is overwritten with the actual image. The main advantage of this method is the low memory need and the fast speed. On some conditions, this method can be executed in image reading time. One of these conditions is that the time between pixel arrivals must be greater than the processing time for a pixel. The processing time consists of one absolute differencing and one addition operation. In most cases, this method can be executed real-time. Unfortunately, this method is highly sensitive to lighting changes, because it makes a hard assumption, that the high intensity difference is motion. The SAD method analyzes each pixel value independently, accordingly global lighting changes are considered as motion. If there is a sudden and/or significant change in illumination, the SAD value can be high enough to be commensurable with an elemental, real motion. This makes impossible to choose a sufficient threshold value. Because of the high speed and low memory need, SAD is suitable for motion detection but only if we can reduce the sensitivity to lighting change. Several types of the SAD method can be found in the literature. Hernández-García at al. presented an extended method using Hamming distances.[5]

Another image processing principle is the estimation of the optical flow. Technically, it is the calculation of movement of the image pixels. The result of these methods is a vector field which represents the motion (velocity) of image pixels. This vector field shows how we can get the actual image from the previous one. These methods make an assumption, that the intensity of pixels is constant and only the location of the pixels can change. Formally

$$I(x(t+\Delta t), y(t+\Delta t), t+\Delta t) = I(x(t), y(t), t) \quad (3)$$

where $I(x, y, t)$ is the image function in time t and Δt is the elapsed time between the two images. This equation called optical flow constraint. Due to the indecision, other constraints must be introduced to estimate optical flow. Several methods are available in the literature, like Horn and Schunck or Lucas and Kanade methods [7], but other differential, correlation-, frequency- and phase based techniques can also be found.[8] The advantage of these methods is that only pixels belong to a moving object is recognized as motion. This reduces the probability of false detections. The main disadvantage of these methods is the complexity. Although there is a possibility of parallel optical flow processing using programmable logic [9], processing using a simple microcontroller could be a time consuming task. Zheng at al. combined background modeling and optical flow processing in one method.[10]



Figure 1: TOF camera

Motion detection methods, which measure spatial arrangement of the scene are also belong to camera based techniques. For instance, Wang at al. developed a stereo vision system [11] as well as Penne at al. presented a TOF (Time Of Flight) camera based system (Figure 1). [12] Kimura at al. developed a new VLSI sensor, which uses a quasi 2D method (projection) to detect motion. [13]

3. Motion detection using low resolution images

Speed of image processing not only depends on the pixel reading time but the resolution of the images. The higher the horizontal or vertical resolution is, the more operations we have to do. Reducing of resolution is not certainly satisfying solution for speed up the detection under the

same efficiency requirements. The resolution reduction causes that the far, small motions become undetectable.

Using low resolution image sequences the detection speed can be raised but far motions detectability is lower. This effect is caused by the operating principle of cameras: the pixel intensity is the mean of light intensity from the subspace belongs to the pixel.



Figure 2: High and low resolution images

The resolution reduction means the enhancement of the subspace of each pixel. The motion in one of these subspaces can cause only the intensity change of the pixel. Therefore, the motion detection algorithms above, particularly the optical flow, become inappropriate solution. The SAD method for low resolution images keeps the disadvantage, that the intensity difference between two sequential image can not only be caused by motion but global illumination change. For low resolution images, it can be ideal to check somehow, if the intensity change caused by motion. This criteria – like the optical flow – can only be estimated.

4. WSSAD method

The basic idea is to introduce a constraint: illumination change causes intensity change of pixels in a whole segment of the image. Accordingly, the pixel intensity change caused by motion if this change significantly differs from the brightness change of the surrounding pixels. We can introduce a method based on this principle, called Window Scaled SAD (WSSAD). This method extends SAD with the constraint above, to distinguish motion from illumination change.

The image is divided to equal size windows. This size can be represented by the integer value R. So thus, the window width (w) and height (h) is:

$$w = \frac{W}{R} \quad h = \frac{H}{R} \quad (4)$$

where W is the width and H is the height of the image in pixels.

The WSSAD algorithm

1. Calculate average brightness of each window of the actual and the previous image in the sequence:

$$AB_{P_{i,j}} = \frac{1}{wh} \sum_{y=(j-1)h+1}^{jh} \sum_{x=(i-1)w+1}^{iw} I_{P_{x,y}} \quad (5)$$

$$AB_{A_{i,j}} = \frac{1}{wh} \sum_{y=(j-1)h+1}^{jh} \sum_{x=(i-1)w+1}^{iw} I_{A_{x,y}} \quad (6)$$

where $i=1..R$ and $j=1..R$.

2. Calculate the WS sum for each window. Sum the pixel intensity change scaled with the brightness change of the window. The SDPX value – the scaled difference of the intensity change – can be calculated with the following formula. Here, dx and dy means the position of the pixel in the window and $x=(i-1)w+dx$, $y=(j-1)h+dy$.

$$SDPX_{x,y} = \begin{cases} I_{P_{x,y}} - \frac{I_{A_{x,y}} AB_{P_{i,j}}}{AB_{A_{i,j}}} & \text{if } AB_{A_{i,j}} \neq 0 \\ I_{A_{x,y}} - \frac{I_{P_{x,y}} AB_{A_{i,j}}}{AB_{P_{i,j}}} & \text{if } AB_{P_{i,j}} \neq 0 \\ I_{A_{x,y}} - I_{P_{x,y}} & \text{otherwise} \end{cases} \quad (7)$$

where $i=1..R$ and $j=1..R$. A T_{pixel} threshold value can be given to get SDPXT values for each pixel:

$$SDPXT_{x,y} = \begin{cases} SDPX_{x,y} & \text{if } SDPX_{x,y} > T_{pixel} \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

With this, the WS value can be calculated using the following formula.

$$WS_{i,j} = \sum_{y=(j-1)h+1}^{jh} \sum_{x=(i-1)w+1}^{iw} SDPXT_{x,y} \quad (9)$$

3. Calculate the GS global sum value. A T_{window} threshold value can be given to reduce noise effects (WST). The GS value is the sum of the

window sums:

$$GS = \sum_{j=1}^R \sum_{i=1}^R WST_{i,j} \quad (10)$$

4. Result. A global threshold value (T_{global}) can be given:

$$R = \begin{cases} MOTION & \text{if } GS > T_{global} \\ NO\ MOTION & \text{otherwise} \end{cases} \quad (11)$$

The selection of the T_{global} , T_{window} , T_{pixel} can be empirical, adaptive or learning algorithms can be used. These parameters of the WSSAD algorithm can be adapted to lighting in the observed area and parameters of the camera.

It should be noted, that the WSSAD method in this form requires storing at least the actual and the previous images in the sequence, because step 1. and step 2. can not be contracted.

5. Implementation proposals

In this section, we discuss implementation problems and proposals for motion detection from low resolution video sequences using a single microcontroller system.

The presented WSSAD method requires storing at least two images from the sequence to detect motion. If the internal memory of the controller allows of storing both images, we recommend to choose this, because of the shorter access time. If it is not possible, the images can be saved in an external storage (RAM for instance). Attached this to the controller, image data become accessible.

If we would make high resolution images in case of motion, we can speed up the method by always taking the high resolution image independently of motion is detected or not. With this extension, the time between the motion and sending the high resolution image can be reduced. Most cameras have snapshot function. Using this function, we do not have to save the image, because the camera module stores it until the presence of motion is determined. If the decision is NO_MOTION the snapshot should be dropped. Another criteria for keeping the high resolution snapshot can be the output of the SAD method. This can be executed in image reading time if there is enough time between pixel arrivals. This time consists of one pixel reading, one addition and one absolute differencing operation time beyond the WSSAD pixel operation time. This means that the high resolution image is kept and the WSSAD is processed only if the SAD detects motion. This extension can efficiently reduce the processing overhead.

If the time between pixel arrivals and the memory size allow, more optimizations of the WSSAD method we can give. The main idea is to calculate the AB values (WSSAD step 1.) in image reading time. This extension needs storing two RxR matrices. Each cell of the matrices have to store a sum of pixels in a window (wh). Using these temporary variables, the WSSAD step 1. can be transformed as follows:

1. Calculate average brightness of each window for the actual and the previous image in the sequence:

```

Let n=1
While n<3
  Let row=0, col=0 and wrow=0, wcol=0
  While wrow<R
    Wait until a PX pixel arrives
    Store PX
    ABn[wcol][wrow]=ABn[wcol][wrow]
    +PX
    If col=h-1 and row=w-1 then divide
    ABn[wcol][wrow] with wh
    Recalculate indices
  End while
  n=n+1
End while
    
```

Here, n is the image number, row and col is the row and column of the actual pixel, wrow and wcol are the window indices. The recalculation of the col, row, wcol, wrow values is as follows:

```

Let col=col+1
If col=w then
  Let col=0, wcol=wcol+1.
  If wcol=R-t then
    Let wcol=0, row=row+1.
    If row=h then let row=0,wrow=wrow+1
  End if
End if
    
```

Accordingly, the additional processing time of the extended WSSAD method step 1 consists of the time of four indirect accessing of the matrix, one division, two condition evaluation and the time of recalculating the indices.

6. Experimental system and results

Our experimental system is a motion sensor consists of an OEM camera, a wireless communication module and a processing module with a microcontroller.

The camera module is connected to the processing

module via serial interface (UART). We use two resolutions:

- 80x60 pixels, 8bit grayscale, Extended BMP format (low resolution image)
- 640x480 pixels, RGB, JPEG format (high resolution image)

The communication module is a ZigBee wireless module in our experimental system.

The processing module is a unique microcontroller module, developed at the Bay Zoltán Foundation for Applied Research (BAY-IKTI), Hungary. It contains a low power consumption microcontroller, an SPI RAM module, and serial lines to the camera and the ZigBee module.

The methods tested using the four most significant test cases:

1. No motion, no lighting change
2. Motion, no lighting change
3. No motion, lighting change
4. Motion and lighting change

To test methods using these cases we made several image sequences. We compared the results of WSSAD and SAD methods. For each test cases, on the following figures, we illustrate the results with the two sequential source images (upper row), the result of the SAD (lower row, left) and the WSSAD (lower row, right). On result images the bright pixel represents motion detected by the algorithm.

In first test case except for the camera noise both the SAD and the WSSAD gave good results. False positive detections could be eliminated with suitable threshold values (Figure 3).

In second test case both methods gave low false negative detection rate with suitable threshold values (Figure 4). For the SAD method it is difficult to choose the threshold value because the deviation of the SAD values is high.

The third case is the most interesting. In this case, the SAD method, due to the lighting change, gave ten times higher values than the WSSAD. This value is commensurable with the values in case of motion. It makes impossible to find an ideal threshold value for the SAD. The false detection rate for this case hits the 21 percent, which is not acceptable. The WSSAD method extremely reduced the effect of lighting change using suitable threshold values (Figure 5).

The last case is the combination of the second and third case. The SAD detected motion because not only of real motion but lighting change. The WSSAD method gave right decision for each sequence. This refers to motion because the WSSAD reduces lighting changes (third test case). The pixels of the result image is only bright near the real motion (Figure 6).

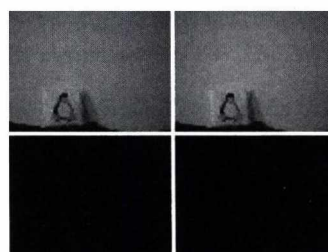


Figure 3: First test case: No motion, no lighting change

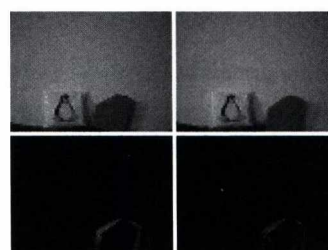


Figure 4: Second test case: Motion, no lighting change

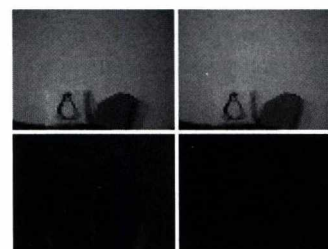


Figure 5: Third test case No motion, lighting change

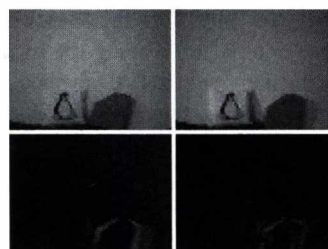


Figure 6: Fourth test case: Motion and lighting change

This system is in test phase at the moment. The robustness of the final system is expected to be higher, and we would like to analyze the possibilities of sensor fusion.

7. Conclusions

In security and home care applications, sensors, which measure environment parameters, plays more and more important role. These sensors set up sensor networks and send data through wireless connection. The goal of this project is to develop a low power consumption motion sensor with the ability to be a node of a wireless sensor network. Using a microcontroller platform, real-time image processing of high resolution is not solvable. The traditional motion detection methods like SAD are not suitable for low resolution, so as an extension, we introduced the WSSAD method. This method calculates the absolute difference scaled with the average brightness change of the surrounding pixels. Before the tests, we analyzed implementation problems, possibilities and made proposals to enhance robustness, reliability and speed of the WSSAD method. For testing, we used an experimental system consists of a camera-, a processing- and a wireless communication module. Experimental tests gave promising results for sequences of each test cases. WSSAD method reduced the effect of sudden lighting change, and only the real motion was detected.

For further work, we would like to connect an infrared device and analyze the further possibilities of sensor fusion in motion detection. Reducing the power consumption is also a future work. We can use sensor fusion, lower power consumption camera and controller to reach this goal. In future we would like to extend WSSAD method using the examination of the effect of shadows on WSSAD method.

References

1. Digital Image Processing , Rafael C. Gonzalez, Richard E. Woods, Prentice Hall, New Jersey, 2002, ISBN 0-201-18075-8
2. An Edge-Based Approach to Motion Detection Angel D. Sappa and Fadi Dornaika, Computer Vision Center Edifici Campus UAB, 08193 Barcelona, Spain, 2006
3. Robust techniques for background subtraction in urban traffic video, S-C.S. Cheung and C. Kamath, IEEE Int. Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance, Nice, France, 2003
4. Optimal Dominant Motion Estimation Using Adaptive Search of Transformation Space, Adrian Ulges, Christoph H. Lampert, Daniel Keysers and Thomas M. Breuel Springer Berlin / Heidelberg, 0302-9743, 2007
5. Video Motion Detection Using the Algorithm of Discrimination and the Hamming Distance, Josué A. Hernández-García, Héctor Pérez-Meana, and Mariko Nakano Miyatake, ISSADS 2005, LNCS 3563, 2005.
6. A General Model for Visual Motion Detection, Takashi Nagano, Makoto Hirahara, Wakako Urushihara, Biol. Cybern. 91, 99–103, 2004
7. Determining Optical Flow, Berthold K.P. Horn and Brian G. Schunck, ARTIFICIAL INTELLIGENCE, MA 02139, Cambridge, USA
8. A phase-based approach to the estimation of the optical flow field using spatial filtering, T. Gautama and M. Van Hulle, IEEE Transactions on Neural Networks, vol. 13, pp. 1127.1136, 2002.
9. FPGA-based Real-time Optical Flow Algorithm Design and Implementation, Zhaoyi Wei, Dah-Jye Lee and Brent E. Nelson, Provo, Utah, USA, 2007
10. Fast Motion Detection Based on Accumulative Optical Flow and Double Background Model, Jin Zheng, Bo Li, Bing Zhou, and Wei Li, CIS 2005, Part II, 2005.
11. Motion Detection in Driving Environment Using U-V-Disparity, Jia Wang, Zhencheng Hu, Hanqing Lu, and Keiichi Uchimura, ACCV 2006, LNCS 3851, 2006.
12. Robust real-time 3D respiratory motion detection using time-of-flight cameras, Jochen Penne, Christian Schaller, Joachim Hornegger and Torsten Kuwert, Int J CARS, 3:427–431 ,2008
13. A Simple-Architecture Motion-Detection Analog VLSI Based on Quasi-Two-Dimensional Hardware Algorithm, Hiroe Kimura and Tadashi Shibata, Analog Integrated Circuits and Signal Processing, 39, 225–235, 2004

Experiments towards an on-vehicle omni-directional camera-based road-safety assessment system

Z. Fazekas, P. Gáspár and A. Soumelidis

Computer and Automation Research Institute,
Budapest, Hungary

Abstract

An on-vehicle omni-directional camera-based road-safety assessment system and approach was outlined recently for narrow, but busy roads. Some experiments pertaining the mentioned system and approach are presented here. The outline of the system is also given.

Categories and Subject Descriptors (according to ACM CCS): I.2.10 [Artificial intelligence]: 3D/stereo scene analysis, I.2.10 [Artificial intelligence]: Video analysis, I.3.3 [Computer graphics]: Viewing algorithms, I.4.8 [Image processing and computer vision]: Tracking and I.4.9 [Image processing and computer vision]: Applications

1. Introduction

Vision-based assessment of road safety is a current topic in the traffic safety literature, see e.g.,⁵. Working along this line of research, an *on-vehicle omni-directional camera-based road-safety assessment system*¹⁴ was outlined for a certain category of roads. Some of the roads that belong to this road category within Hungary are critical in the context of the country's traffic safety.

Some *initial results* from the *experiments* pertaining to the aforementioned system and approach are presented herein. Also, the *outline of the proposed system* is presented in the paper.

1.1. Roads and accidents

Big volumes of road accident *statistical data* are available in respect of the *EU* and *OECD countries*¹⁵. Comprehensive studies analyze the *causes, circumstances* and the *fatalities* of these accidents, as well as, the perceptible *trends* behind these. Other studies estimate the direct and indirect *economic costs*⁶ arising from these accidents.

In Hungary, *fatal traffic accidents* number about one-thousand per year with about seven-thousand traffic accidents per year resulting in *serious injuries*.

With only a relatively small portion of the budget being spent on constructing new *motorways*, and on the mainte-

nance and reconstruction of the existing *road infrastructure* – which is as in a fairly poor shape already – it is of great importance to use the available funds in a meaningful way.

1.2. R&D stanchions

We build upon *several stanchions of the R&D work* in developing a *viable system* and approach. These are the experience gained in some of our earlier R&D projects and the information published by other teams.

Our own experience covers *computer vision based traffic lane detection and tracking*¹³, and *modelling the light-reflection at free-form specular surfaces*, as well as, the *geometrical reconstruction of specular surfaces*⁹ – e.g., the shape of the tear-film coated human cornea – from a single view or multiple views.

It is probably worthwhile to explain *how* and *in what sense* these experiences are *relevant* to the development of vision-based road safety assessment system. First of all, the *concrete application field* and the *sensing modality* are common with those associated with the traffic lane tracking system mentioned above.

The *algorithms*¹⁶ and *tools*⁸ developed in conjunction with the mathematical modelling of specular surfaces are expected to be particularly useful in the development of the proposed system. This is because non-spherical – in

cases custom-designed – mirrors are often used in catoptric and catadioptric omni-directional systems. So, for instance, a *custom-shaped mirror* can be derived – i.e., geometrically specified – from a rough specification of the omni-directional system's expected optical behaviour with the help of these algorithms and tools. Some further details are given in Section 4.

We had to rely on external sources of information in respect of the *surrogate safety measures*², the *Surrogate Safety Assessment Model (SSAM)*¹¹, as well as, in respect of the *omni-directional vision systems*^{4,10}.

Building on these stanchions, a vision-based approach of *assessing traffic safety on narrow, but busy roads* was proposed in¹⁴. The imaging device suggested there was a catadioptric omni-directional camera mounted on a probe-vehicle (i.e., a vehicle that moves with the road traffic).

2. Traffic surveillance approaches

In the present paper, the *traffic safety related aspects of traffic surveillance* are emphasized, though, clearly, such systems are used for other traffic related purposes, as well. These purposes include e.g., *traffic management*, and *travel time collection*. The collected data could then be sent either in an offline, or an online manner to data collecting centres. The mode of data management obviously affects the way the collected data can be utilized. In case of the road safety assessment, *both real-time and offline implementations* could be meaningful. The real-time implementation, however, could support a richer set of road safety-related functionalities.

There are *two main types of traffic surveillance systems*, namely road-based and vehicle-based systems. The *road-based traffic surveillance systems*, such as *inductive loops*, have been the most important means of road surveillance and incident detection for many years. It was mainly due to their *reliability and inexpensive operation*.

With the fast and significant changes experienced in sensor technology⁷, the *camera-based* and other recent road-side vehicle detection technologies are now also used extensively in traffic measurements, e.g., for measuring high-volume road traffic^{3,10}. Simultaneously with the technological advances concerning vehicular sensors, the increased speed and reliability of the *detection algorithms* has also significantly eased the *implementation efforts* involved in the development of vehicle-based surveillance systems.

Vehicle-based traffic surveillance systems involve *probe-vehicles* – equipped with some tracking devices – that allow the vehicles to be tracked by a central computer facility. Although, vehicle-based traffic surveillance systems are not yet in wide use, these systems could *gather rich data on travel times* and *detect traffic incidents*. They can also be used to estimate traffic flows and origin-destination patterns.

Recently, a *visual vehicle-based traffic surveillance system*, interestingly, but not surprisingly *using an omni-directional vision approach*, was proposed for the above outlined applications⁴. From our point of view, the paper's most relevant contribution was the definition, generation and the use of *dynamic panoramic surround maps*, as these maps could be used to calculate the *surrogate safety measures* given in².

2.1. The case of busy and narrow roads in Hungary

Though *some sections of motorways and multi-lane roads* have been built in the last few years in Hungary, a *significant percentage of the road traffic* is served by two-lane – i.e., *one-plus-one-lane* – roads. Unfortunately, still lengthy sections of the *national main road network* belong to the one-plus-one-lane road category. Some of these sections – e.g., sections along the national main road no. 4 – are burdened with *intense and relatively slow truck and lorry traffic*. These sections tend to be *over-saturated* for longer and longer peak-hours. The situation is saddened by the *impatience and carelessness* of drivers. See the illustrative snapshot shown in Fig. 1.

Indeed, the *drivers' irresponsibility* has reached an astonishing and frightening level. *Manoeuvres* – such as risky and life-threatening overtakings – have become the *"norm"* on these one-plus-one-lane roads. In many cases, it is *not the speed-limit* though that is violated, but the prohibition, or the *safety rules of overtaking*. The all the factors render these sections *extremely unsafe*. We will refer these roads as *target roads*.

There is a growing governmental intention to *monitor, regulate and control* the traffic on these roads, e.g. by *deploying speed cameras* and other traffic surveillance equipment,



Figure 1: Lengthy sections of the Hungarian main road network belong to the one-plus-one lane road category. Some of these sections are extremely unsafe. It seems that the drivers' irresponsibility has reached an astonishing and frightening level.

and by using *traffic-dependent traffic light control regimes* that optimise the throughput of junctions. However, there is still much to be done in respect of accident prevention on the target roads.

In view of this poor traffic safety situation, it is *questionable* whether viable measurements and effective preventive measures, in respect of the target roads, with any hope of success *can be achieved by solely stationary means*. It seems to us that *at least some* measurement devices should *move together with the traffic*. In other words, apart from the stationary (i.e., road-based) surveillance approaches and devices already widely used, the use of some sort of vehicle-based surveillance approach and system seem to be necessary. In ¹⁴, we suggested using a *mobile visual* vehicle-based traffic surveillance system – somewhat similar to the one described in ⁴ – for the target roads.

The *data collected* in this manner could be processed, analyzed and *data-mined* with the aim of identifying *non-trivial unsafe geographic locations* – e.g., road sections other than junctions – and *other conditions* – e.g., weather, working day/weekend – where and under which unusual vehicular and driver behaviour frequently occur. This can be done in a similar manner to the validation of the crash-prediction capability of micro-simulations of intersections reported in ². This way, potential nontrivial accident hot-spots could be identified and appropriate preventive actions could be taken.

3. SSAM for dynamic traffic scene evaluation

Traditionally, *transportation agencies* had used statistical *analysis of crash records* to evaluate the safety of intersections, interchanges, and other traffic facilities. If a specific location had "produced" high number of crashes over years, this location was investigated and possible remediation was considered. Unfortunately, this process assumes an *excessive number of crashes* to have happened at locations.

Although this method will stay with us to calculate the grim ground truth about crashes, as a less gloomy alternative, *crash-number prediction formulas* based on traffic-flow values were proposed as early as 1960. At about the same time, with similar motivation, that is to assess and manage the safety of traffic facilities more effectively and in a preventive fashion, the use of *surrogate safety measures* was proposed. The surrogate traffic safety assessment evaluates *traffic conflicts* – scenarios where two or more road users are likely collide without evasive action taken – *instead of crashes* in the given location.

The *basic assumption* of the surrogate traffic safety assessment is that *conflict frequency* is correlated with the *risk of actual collision*. (The conflicts considered in the safety assessment can be real or simulated.)

A major model based on this approach – *Surrogate Safety*

Assessment Model (SSAM) ¹¹ – was developed and validated in the USA.

SSAM augments the analysis of the expensive and unreliable *field measurements* and *actual crash* data and uses runs of *appropriately tuned micro-simulations* of the road traffic at junctions or road segments to *assess traffic safety* in these locations.

Though, some traffic *volume-based accident-prediction models*, e.g., ¹, show better correlation with factual crash data than the SSAM-based predictions, the micro-simulation-based models provide a good insight into the underlying traffic patterns and accident types. The importance of micro-simulation-based models is evident in case of *uncommon road geometries* and arrangements, and in case designs, i.e., junctions not yet built.

Furthermore, there is a *wide range of differences* between countries in the usual/local behaviours on roads. Micro-simulation-based modelling seems to be more tuneable toward these local behaviour patterns.

As declared in ¹¹, the SSAM is expected to be used on *real data acquired from visual traffic surveillance systems*. Since the camera coverage of the busiest intersections in Hungary increase, the surrogate safety measures could be collected via automated video processing methods at least for research and experimental purposes in the future.

Note that in case of vehicle-based traffic safety surveillance opted for in this paper, the surrogate safety measures need to be *adapted to non-stationary measurements*, i.e., when also the ego-motion of the vehicle hosting the surveillance camera must be considered.

4. Experiments with catadioptric omni-directional imaging arrangements

One of the stanchions of development mentioned in Subsection 1.2 was the experience gained in the *mathematical modelling light-reflection at free-form specular surfaces*. This is because non-spherical – possibly custom-designed – mirrors are often used in *catoptric* and *catadioptric* optical systems in general, and omni-directional systems in particular.

4.1. Specular surface reconstruction method used in free-form mirror design

The specular surface reconstruction from a view can be achieved by the integration of *partial differential equation (PDE)* derived from the *mapping* between the *measurement-pattern* and its *virtual image* taken by the camera ^{9, 16}. In the present application context, similar mappings can be created considering the *expected optical behaviour* of the imaging system, then the corresponding PDE's can be numerically integrated for practically acceptable settings. A *smooth specular convex surface F* can be described in preferably chosen

spatial coordinate system, for instance, in that of fixed to the camera. The origin B of the coordinate system can be conveniently placed in the optical center of the camera, while its z -axis could coincide with the optical axis BB' of the camera. The specular surface F is then described in the following form:

$$F(x_1, x_2) = S(x_1, x_2)\hat{x} \quad (\hat{x} = (x_1, x_2, 1)^T) \quad (1)$$

In the above equation, $S(x)$ ($x = (x_1, x_2)$) is a scalar-valued function describing the inverse distance-ratio, measured from B , of the 3D point P_x – corresponding to \hat{x} – and the surface-point appearing in the same direction as P_x from B .

The propagation of light from the points of the measurement pattern to those of the distorted image, i.e., $P_y P P_x$, is described in this coordinate system. By doing so, a mapping is identified between the points P_y of the measurement pattern and the points P_x of the image: $P_y \rightarrow P_x$. The mapping $P_y \rightarrow P_x$ – under the given circumstances – is a one-to-one mapping. By the physical law of light-reflection, $S(x)$ – describing surface F according to (1) – must satisfy the following first-order PDE:

$$\frac{1}{S(x)} \frac{\partial S(x)}{\partial x_j} = \frac{v_j(x) - x_j}{\langle \hat{x}, \hat{x} - v(x) \rangle} \quad (j = 1, 2), \quad (2)$$

where

$$v(x) = |\hat{x}| \frac{k + f(x) - S(x)\hat{x}}{|k + f(x) - S(x)\hat{x}|},$$

and function $f(x)$ can be expressed with the inverse of $P_y \rightarrow P_x$ mapping, i.e., with mapping $P_x \rightarrow P_y$. In (2), k is a vector pointing to a reference point in the plane of the measurement pattern; while $\langle \cdot, \cdot \rangle$ denotes the scalar product of the 3D space.

The surface F can be determined uniquely under the starting condition of $S(0,0) = s_0$, if the $P_y \rightarrow P_x$ mapping is known. By applying the algorithm given in ¹⁶ to the above PDE for different starting conditions, candidate free-form surfaces can be derived and considered for the physical realization.

The developer of the imaging system needs to take into consideration the costs of manufacturing the mirror and its installation on the host vehicle, as well as the acceptable size of the optical system. By doing so, some viable shape-specifications can be obtained. If only surfaces-of-revolution mirrors – attractive from an economic point of view – are considered, then (2) can be further simplified.

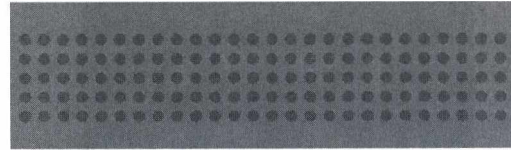


Figure 2: Test image no. 1 – it is used for identifying a horizontal plane about 1.5 m above the horizontal road surface.

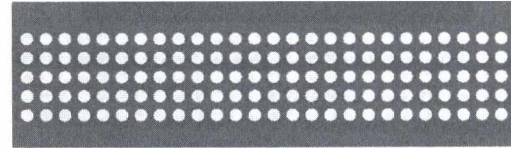


Figure 3: Test image no. 2 – it is used for identifying the horizontal road surface.

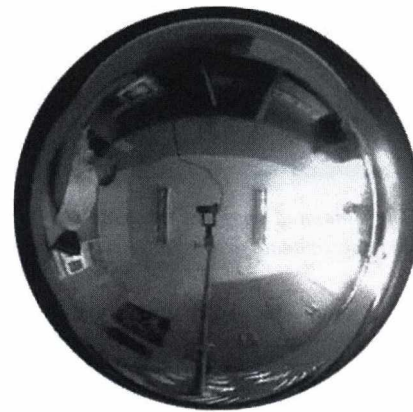


Figure 4: An omni-directional snapshot in a lab taken with a catadioptric imaging system comprising a web-camera and hemispherical mirror.

4.2. An omni-directional arrangement for road safety assessment

The advantage of using omni-directional camera in the given context can be verified by considering that eight or more synchronized conventional cameras would be necessary to cover the 360° field-of-view – required in this dynamic traffic safety assessment task – in a monocular fashion. To cover this field-of-view in a stereoscopic fashion – i.e., doubly to enable the computation of point correspondences – even higher number of cameras would be necessary. Using a fish-eye stereo vision system as suggested in ¹⁰, still at least four synchronized cameras would be necessary. Note that the cameras would need separate mountings, connections, and possibly separate calibration procedure.

Various omni-directional vision systems are known from the literature. The dioptric systems and catoptric systems

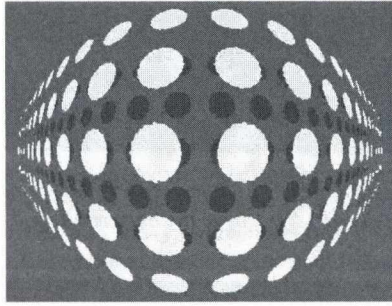


Figure 5: The simulated image of two planes with circular blobs.

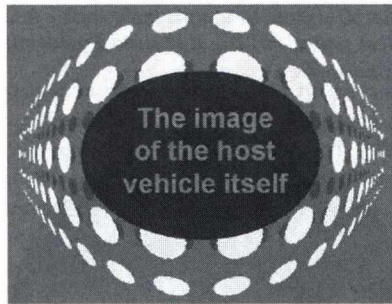


Figure 6: The simulated image of two planes with circular blobs with the approximate area of the host vehicle marked.

are based on special lenses and on non-planar mirrors, respectively; while the *catadioptric* systems use both of these optical elements. Some of these systems – including omni-directional traffic surveillance systems – are reviewed in 4.7.12. By using omni-directional optical systems, one can expect non-customary, *strongly distorted images*. Such images are shown in Figs. 4 and 5.

In our experiments, two simple spotty *test patterns* were used to represent the *road surface* and another horizontal plane about 1.5 m above the road surface. These test patterns are shown in Figs. 2 and 3.



Figure 7: The office scene of Fig. 4 is flattened into the plane of the ceiling.

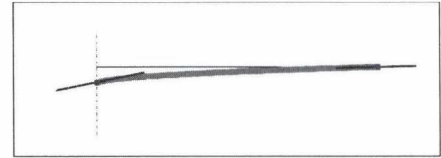


Figure 8: Sketch of a specular surface of revolution – with central apex – for a parameter setting.

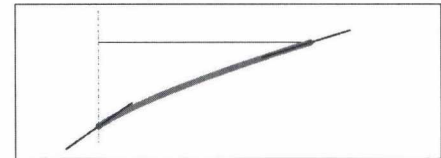


Figure 9: Sketch of a specular surface of revolution – with central apex – for another parameter setting.

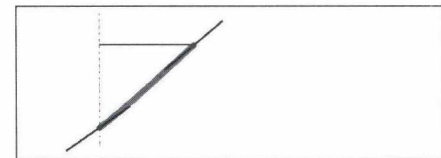


Figure 10: Sketch of a specular surface of revolution – with central apex – for yet another parameter setting.

In a *possible arrangement* of a catadioptric vision system for the given purpose, an *upward looking conventional camera* and a *non-planar mirror* – e.g., hemispherical mirror – are rigidly mounted on a *rod* installed on the top of the host car. The *mapping* – generated via reflection of light at the surface – can be perceived via comparing in Figs. 2, 3 and 5.



Figure 11: A surface-of-revolution mirror with a central apex.

In Fig. 6, the approximate area, a fairly big portion of the image, that *reflects the host vehicle* – and therefore irrelevant for the detection of the road environment and traffic – has been marked.

By choosing some appropriate *non-spherical specular surface* – such as shown in Fig. 11 – for the catadioptric vision system, a *better use of the image sensor area* can be achieved. Figs. 8, 9 and 10 show the *approximate shapes* of the mirrors for *various camera parameters and settings* that corresponding to various starting conditions and result in *mirrors of various sizes*.

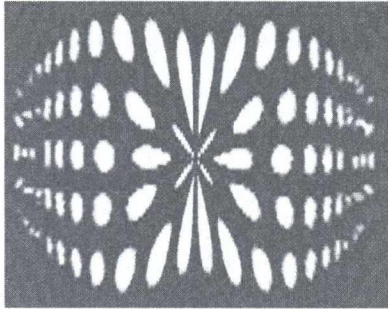


Figure 12: The qualitative effect of using a mirror with central apex – such as shown in Fig. 11 – in the catadioptric imaging system.

In Fig. 13, an improved catadioptric arrangement – based on a specular surface-of-revolution with central apex – is shown.

The qualitative camera view is shown for such an arrangement in Fig. 12.



Figure 13: Sketch of a catadioptric system – comprising an upward-looking camera and a rotational symmetrical non-planar mirror with an apex – mounted on a host vehicle. The camera and the mirror are rigidly mounted on a rod.

4.3. Processing carried out in the system

Though, the main processing steps of the vehicle-based traffic safety assessment task are not markedly different from the general scheme of the lane detection algorithms – given in ¹³ – based on cameras and on additional sensors, some additional processing steps must be included in the mentioned scheme.

From the image data – taken by the omni-directional imaging arrangement – the road and vehicle features are extracted. Based on these features and considering the road surface model – e.g., planar road – and the sensor model – i.e., the model describing the rigid arrangement of a known specular surface and a calibrated camera – the outliers are

removed from the road features. Then a lane model is fitted on the blobs representing the inlier road features.

Then taking ego-vehicle dynamics into account, tracking is carried out with respect to the road and – after proper filtering – to other vehicles. Finally, the surrogate safety measures are calculated based on these tracking results.

5. Conclusion

In this paper, we presented some initial experimental results that could facilitate the development of the on-vehicle omni-directional camera-based road-safety assessment system – proposed in ¹⁴ – for narrow, but busy roads.

In this particular application, both online and offline data collection make sense. In order to assess the safety of the road traffic, traffic events – called incidents – must be extracted from the image sequences and categorized according to the SSAM methodology.

The use of a single omni-directional optical system with an on-board computer equipped with appropriate video, navigational and vehicle dynamics data recording capabilities is foreseen to be appropriate for the purpose.

Acknowledgements

The financial support of this work provided by the Hungarian National Office for Research and Technology – in the frame of grant TECH_08_2220080088 – is gratefully acknowledged by the authors.

References

1. C. V. Zegeer, J. Hummer, L. Herf, D. Reinfurt and W. Hunter. *Safety Effects of Cross-Section Design for Two-Lane Roads* Report No. FHWA-RD-87-008, Federal Highway Administration, Washington, D.C., USA, 1986. 3
2. D. Gettman and L. Head. Surrogate Safety Measures from Traffic Simulation Models. *Transportation Research Record: Journal of the Transportation Research Board*, (1840), pp. 104–115, 2003. 2, 3
3. S. K. Gehrig. “Large-Field-of-View Stereo for Automotive Applications”, *Sixth Workshop on Omni-directional Vision, Camera Networks and Non-classical Cameras*, Beijing, People’s Republic of China, (2005). 2
4. T. Gandhi and M. M. Trivedi. Vehicle Surround Capture: Survey of Techniques and a Novel Omni Video Based Approach for Dynamic Panoramic Surround Maps. *IEEE Transactions on Intelligent Transportation Systems*, 7(3), pp. 293–308, 2006. 2, 3, 5

5. N. Saunier, T. Sayed and C. Lim. "Probabilistic Collision Prediction for Vision-Based Automated Road Safety Analysis", *IEEE Intelligent Transportation Systems Conference*, Seattle, Washington, USA, pp. 872–878, (2007). 1
6. V. Sesztakov and Á. Török. "Monetary Value of Road Accidents in Hungary", *International Scientific Conference on Modern Safety Technologies in Transportation*, Kosice, Slovakia, pp. 250–253, (2007). 1
7. M. M. Trivedi, T. Gandhi and J. McCall. Looking-In and Looking-Out of a Vehicle: Computer-Vision-Based Enhanced Vehicle Safety. *IEEE Transactions on Intelligent Transportation Systems*, **8**(1), pp. 108–120, 2007. 2, 5
8. A. Bódis-Szomorú. *Cornea Specular Reflection Simulator (Users Manual)*. MTA SZTAKI, Budapest, Hungary, 2008. 1
9. Z. Fazekas, A. Soumelidis, A. Bódis-Szomorú, and F. Schipp. "Specular Surface Reconstruction for Multi-camera Corneal Topographer Arrangements", *30th Annual International Conference of IEEE EMBS*, Vancouver, BC, Canada, pp. 2254–2257, (2008). 1, 3
10. S. K. Gehrig, C. Rabe and L. Krueger. "6D Vision Goes Fisheye for Intersection Assistance", *Canadian Conference on Computer and Robot Vision*, Windsor, Ontario, Canada, pp. 34–41, (2008). 2, 4
11. D. Gettman, P. Lili, T. Sayed and S. Shelby. *Surrogate Safety Assessment Model and Validation: Final Report FHWA-HRT-08-051*. Federal Highway Administration, Washington, D.C., USA, 2008. 2, 3
12. J.-F. Layerle, X. Savatier, M. El Mouaddib and J.-Y. Ertaud. "Catadioptric Vision System for an Optimal Observation of the Driver Face and the Road Scene", *30th AIEEE Intelligent Vehicles Symposium*, Eindhoven, The Netherlands, pp. 410–415, (2008). 5
13. A. Bódis-Szomorú, T. Dabóczy and Z. Fazekas. "A Lane Detection Algorithm Based on Wide-Baseline Stereo Vision for Advanced Driver Assistance", *7th Conference of the Hungarian Association for Image Processing and Pattern Recognition*, Budapest, Hungary, Paper-id. 142-2, pp. 1–10, (2009). 1, 6
14. P. Gáspár, Z. Fazekas and A. Soumelidis. Notes on Vehicle-based Visual Traffic Surveillance for Crash-prediction. *Science & Military*, (2), pp. 26–32, 2009. 1, 2, 3, 6
15. *OECD Factbook 2009*. OECD, 2009. 1
16. A. Somelidis, Z. Fazekas, A. Bódis-Szomorú, F. Schipp, B. Csákány and J. Németh. "Specular Surface Reconstruction Method for Multi-camera Corneal Topographer", in *Recent Advances in Biomedical Engineering*, G. R. Naik (Ed.), In-Teh, Vukovar, Croatia, pp. 639–660, (2009). 1, 3, 4



ISBN 978-963-421-591-2