Stony Brook University

Refraction in Graphics and Medical Imaging

A Dissertation Presented
by
Shengying Li

to
The Graduate School
in Partial Fulfillment of the
Requirements
for the Degree of
Doctor of Philosophy
in
Computer Science
Stony Brook University

August 2008

Stony Brook University
The Graduate School

Shengying Li

We, the dissertation committee for the above candidate for the degree of Doctor of Philosophy, hereby recommend acceptance of this dissertation.

Klaus Mueller, Dissertation Advisor
Professor, Computer Science Department

David Xianfeng Gu, Chairperson of Defense
Professor, Computer Science Department

Jerome Zhengrong Liang
Professor, Departments of Radiology

Lawrence H. Staib
Professor, Departments of Diagnostic Radiology
Yale University

This dissertation is accepted by the Graduate School

Lawrence Martin
Dean of the Graduate School

Abstract of the Dissertation

Refraction in Graphics and Medical Imaging

by

Shengying Li

Doctor of Philosophy

in

Computer Science

Stony Brook University

2008

With the fast advancement of computer graphics hardware, the development of volume graphics in recent years has extented from volume visualization of scientific data to more general graphics technology. This thesis is to fill vacancy of research of natural phenomenon associated with the refraction effect on the visualization of volume data. We have further applied our refraction research to medical imaging to correct refraction distortion for ultrasound computed tomography technology.

First, we find that refraction imposes significantly higher demands on filters for interpolation and gradient estimation than illumination and shading. To firmly support refraction from the perspective of fundamental digital processing, we evaluate the family of spline filters as a good alternative to the cubic filters, which so far served as the gold standard of efficient yet high-quality interpolation filters in present visualization applications. Then we developed an efficient framework for the high-quality rendering of discretely sampled surface-based objects with refractive effects. It combines an accurate estimation of the refraction indices, paired with efficient and accurate surface detection, space traversal and backdrop image sampling. Furthermore, we explored in more general rendering field of graphics by proposing a new concept of geometry field for real-time reflection and refraction using ray tracing. With the combination of the advantage of light fields and geometry images, a geometry field can effectively fetch all possible ray/object intersections with geometry information in fast table lookups. Finally, we applied our research

of refraction to medical imaging. A significant obstacle in the advancement of Ultrasound Computed Tomography has been the lack of efficient and precise methods for the tracing of the bent rays that result from the interaction of sound with refractive media. We propose to find the wave frontier by solving the Eikonal equation which governs the propagation of sound waves. This enables us to determine the ray path with great accuracy. With GPU acceleration, we perform reconstruction with high fidelity and accurate geometry in clinically practical time.

For my husband Weidong and sweet angel Lisa.

Contents

List of Figures

List of Tables

Acknowledgements

I would like to express my deep and sincere thanks to my dissertation advisor, Professor Klaus Mueller, for his insightful inspiration and wisdom, and for teaching me how to do high-quality research. Without his encouragement and guidance, I cannot imagine myself to finish this thesis. He showed such patience and enthusiasm and spent numerous hours to discuss with me, that stimulate my curiosity and strong interests to dig out the research problems at the best of my ability and beyond.

I would also like to thank all my dissertation committees, Professor Lawrence H. Staib, Professor Jerome Zhengrong Liang and Professor David Xianfeng Gu for their precious time to serve in my committee and their invaluable suggestions.

I would like to acknowledge my collaborators and co-authors in the work for this dissertation. Many thanks to Zhe Fan and his adviser, Professor Arie Kaufman, Xiaotian Yin, Professor David Gu, Professor Marcel Jackowski, Professor Lawrence Staib and Donald Dione, for their in-depth discussions and collaborations that I have substantially benefited.

All members in Visualization Laboratory at Stony Brook have contributed to this thesis and I pleasantly thanks the friends and colleagues for sharing with me good time and helping my daily life in Stony Brook enjoyable in the lab: Fang Xu, Lujin Wang, Julia Eun Ju Nam, Wei Xu, Supriya Garg, Erin Golub, Ziyi Zheng, Aili Li, Zhe Fan, Feng Qiu, Susan Frank, Wei Hong, Kyle Hegeman, Miao Jin, Xiaotian Yin, Xin Li, Hongyu Wang, Wei Zheng, Lei Zhang, Wei Zhang.

I would thank NIH grant, NSF grant for partial support of my thesis work.

I would thank my husband, Weidong Jin, for his love. He always gives me his full support and instant help on everything I am engaged in. I would like to thank my parents, Changgui Li and Youfang Hu for their consistent caring and

warm encouragement from the other end of countless international phone calls. I appreciate my sister, Xiao Li, who accompanies with me to grow up and pursue the meaning of life. I also appreciate my little daughter, Lisa Jin, whose arrival gives me strong motivation to create a promising and enjoyable life.

Publications

1. S. Li, K. Mueller, M. Jackowski, L.H. Staib, D. P. Dione, Physical-Space Refraction-Corrected Transmission Ultrasound Computed Tomography Made Computationally Practical, International Conference on Medical Image Computing and Computer Assisted Intervention, New York, to appear in Lecture Note of Computer Science, 2008
2. S. Li, Z. Fan, X. Yin, K. Mueller, A. Kaufman, X. Gu, Real-time Reflection Using Ray Tracing with Geometry Field, in EUROGRAPHICS 2006, Vienna, pp. 29-32, 2006
3. S. Li, K. Mueller, M. Jackowski, D. P. Dione, L.H. Staib, Fast Marching Method to Correct for Refraction in Ultrasound Computed Tomography, in IEEE International Symposium on Biomedical Imaging06, Washington D. C., pp. 896-899, 2006
4. S. Li, K. Mueller, Spline-Based Gradient Filters For High-Quality Refraction Computations in Discrete Datasets, in Eurographics/IEEE VGTC Symposium on Visualization, Leeds, United Kingdom, pp. 217-222, 2005
5. S. Li, K. Mueller, Accelerated, High-Quality Refraction Computations for Volume Graphics, in Volume Graphics, New York, United States, pp. 73-81, 2005
6. S. Li, N. Hatori, Y. Ajima, S. Sakai, H. Tanaka, VLDP Multipath Execution: Mechanism and Evaluation, Information Processing Society of Japan (IPSJ) 2001-ARC-144, Okinawa, Japan, pp. 105-110, 2001

Chapter 1

Introduction

1.1 Refraction

Refraction, in physics, stands for the phenomenon of the deflect of a wave on passing from one medium with a velocity into a second medium with another velocity. In addition to light waves, other forms of waves, such as sound or ultrasound waves, can be refracted too. Refraction plays important roles in our lives. Serving for different purposes, such as in a telescope or microscope, a lens relies on refraction to form images of an object. Our eyes are natural lens. Refraction lets light pass though and generate images for the brain. Refraction is also imperative for colorful rainbows. The material in the air acts like a prism, splitting white light into different spectrum colors. Refraction causes some other interesting optical phenomena and illusions, such as shimmering, mirages and Fata Morgana.

1.2 Problem Statement

The first fundamental question that arises from refraction in computer simulated processing is how to represent the deflection of waves accurately. In the digital world of the computer, geometries are represented by discretely sampled subjects, either with gridded or meshed datasets. Both of them need to perform discrete signal processing. An integral component in the visualization and graphics of discrete data are the filters used for interpolation and derivative estimation.

While the interpolation filter determines the geometric accuracy of the object estimated from the sampled data, the derivative filter affects its perceived accuracy. The latter is due to the derivative's involvement in the shading computation, which appears in its unit-length form as the normal vector. Inaccurate normals will result in the depiction of false structures and patterns on the rendered object surface.

While the modulation of the normal vectors was exploited to generate fictitious patterns in bump mapping [6], this is clearly not intended in a typical visualization where one desires to illustrate the truth. In more recent research, the derivatives have also been used to estimate the geometry itself, by determining the exact position of an isosurface via locating extreme points in the first derivative and zero-crossings in the second derivative [35] [36].

Refraction poses especially high demands on the accuracy of gradient estimation, since it relies on the estimation of gradients to decide the path of the refracted rays. It magnifies any gradient estimation errors by sending the refracted ray along the wrong path, which can potentially lead to more severe visual artifacts than when the normal vector is poorly estimated in illumination calculations. Also, while the gradient is estimated by the density function in the volume, points in the data set are discrete-valued, which may not be able to represent the continuous density function strictly. Thus, even if an analytic function has been sampled into the grid, storing the sample points in the datasets as truncated integers will result in rounding errors. Finally, real-world data sets, such as medical datasets, may be noisy, which will also lead to artifacts in the gradient calculation. The refraction process will accentuate any of these imperfections tremendously. Therefore, a good gradient filter with just the right amount of smoothing and anti-aliasing is strongly recommended for high-quality refraction.

Imaging with refraction suffer from the under-sampling. Due to the deflection of rays, the refracted images with few samples may stand for an extraordinary large area information. When the signal sampling rate is lower than Nyquist sampling rate, the object cannot be represented accurately and aliasing is visible.

Supersampling is a natural solution to the problem of undersampling. It shoots a matrix of rays per pixel, possibly with jitter, and averages the result. Its downside is excessive cost. We propose a new method, named post-refraction supersampling, to save the largest part of the computational cost for the ray tracing while retaining

the supersampling quality.

Interpolation filtering with good anti-aliasing characteristics is another potential approach to attenuate aliasing introduced by a low sampling rate. The goal is to provide a cutoff frequency that removes unwanted signals from the input signals and attenuates them to the degree that then will not adversely affect the result. The key factors of concern are the amount of attenuation in the passband, the rolloff in the stopband and the steepness in the transition region. An ideal filter has a sharp ”wall” response, with an infinitly small transition region. But real world filters are not perfect.

Calculating and tracing intersections between rays and rendered objects is very challenging. In order to render realistic images of refraction in computer graphics, ray tracing is a commonly used method for its high quality and accuracy. However, to calculate the illumination distribution at the object surface, refraction intersection tests between the object and viewing rays is necessary but time-consuming. It is usually one of the most important bottlenecks for performance. Although there has been a lot of research on this area with the goal of improving the speed and keeping the quality using software or hardware, this is still a famous open problem. In the case of color splitting such as by a prism, original white light is diverted into several color rays and each of them needs to do intersection tests with the surface. Applying intersection tests quickly and accurately are the key issues to improve the speed performance of ray tracing.

Playing an imperatively important role in medical imaging, computational tomography reconstructs an object from data collected from different directions either by penetrating the object or reflecting from interfaces. Traditional computational tomography algorithms deal with projections in straight rays, like x-rays, or ”diffracted projection” in phenomena with scattering in practically all directions, such as ultrasound or microwaves in the case that the object interface is smaller than the wavelength. But there is a gap left between these two categories: refraction. Few research efforts are focused on this interesting natural phenomenon. How to correct the refract effect in computational tomography reconstruction without diffraction in all directions?

This is a problem which is not answered thoroughly in research yet. This phenomena is not a rare case and it happens when the wave passes through object

interfaces with a different velocity while the object's size is larger than the wavelength of the wave. Especially in breast ultrasound, the subcutaneously large fat layer has non-ignorable difference from inside materials, which causes strong refraction and heavy distortion. This problem needs to be solved since breast cancer is the biggest killer for women nowadays and medical workers use ultrasound as routinely adjunct to the patient test and treatment. Without a powerful tool to accurately trace the curved ray, the reconstruction result of medical imaging data can be highly distorted and misleading to doctor's diagnosis decision.

In all areas, signal processing, visualization, computer graphics and computational tomography, high quality and speed are hard to gain at the same time. How to accelerate the processing while keeping high quality is a typical problem researchers are trying to solve over all of these years? In more detail, how can we improve the refraction quality while maximally decrease the speed penalty?

1.3 Contributions

To overcome the above difficulties, we have done numerous investigation in the phenomenon of refraction crossing the area of signal processing, computer graphics and computational tomography and record them in this dissertation. In more detail, it includes:

1.3.1 Normal estimation of integration filters

We evaluate the family of spline filters as a good alternative to the Catmull-Rom cubic filters in refraction, which has dramatically higher demands on gradient filters than common illumination and shading. Cubic filters are usually treated as the gold standard of interpolation filters in visualization due to their high-quality and acceptable time cost. Our research indicates that splines can be superior to the Catmull-Rom filters, with potentially less computational overhead, and equal or better quality. Besides, splines provides a powerful and accurate means to adjust the extent of smoothing, which is imperative to minimize numerical error for digital signal process in refractive graphics.

Various researchers have disclosed the importance of the accuracy of gradient

estimation. Gradients estimated with a filter based on a cubic polynomial, such as the Catmull-Rom spline, have been believed to be the unofficial standard for high-quality rendering in visualization. This is widely used in illumination and shading computation. Although higher-order filters tend to generate more accurate derivatives, they require excessive computational expense. Therefore, the Catmull-Rom spline is generally perceived to be a good compromise between computational effort and accuracy, especially when high-quality rendering is the goal.

However, our research indicates that Catmull-Rom spline filters are insufficient for high-quality rendering of refractive effects since here the error is multiplied by the length of the redirected ray before it hits an opaque surface. While the interpolation filter determines the geometric accuracy of the object estimated from the sampled data, the derivative filter affects its perceived accuracy. The derivative involved in the shading computation appears in its unit-length form as the normal vector. In refraction, the normal estimation further dominates the refracted rays direction and position intersected on the background. With ray proceeding, small errors of the unit-length normal vector can be accumulated along the refractive path. Inaccurate normals will result in the depiction of false structures and patterns on the rendered object surface.

With a thorough understanding of the attributes of normal estimation in refractive graphics, we have given renewed attention to the topic of derivative estimation. In our study, the visualization of an object refractive properties was the main goal. We have demonstrated that the B-Spline filter achieves superior results, compared to the traditional Catmull-Rom filter, for the estimation of gradients from discrete data. It does so at the same computational cost for the B-Spline-3, which has the same support, or at $3/4$ of the cost for the B-Spline-2, which has a full spatial support of 3.0. While B-Splines require the computation of coefficients from the raw densities, to be used in the interpolation process, this is only required once as long as the densities are not modified, which is unlikely for pure visualization purposes. We have also demonstrated that the B-spline allows one to balance smoothing with grid sample interpolation fidelity. This is beneficial in the presence of noise, round-off errors, and other artifacts incurred in the sampling of the original data. This is also very important to keep normals in local areas continuous and coherent, which results in a clear pattern in the final rendered surface.

1.3.2 Post-refraction supersampling

We propose a naive method, named as post-refraction supersampling, for ray-tracing in refractive graphics as a method accurately preserve the refractive image patterns, effectively increasing the sampling rate and quality of refracted images while avoid the tremendous computational expense of extra ray tracing for traditional supersampling methods.

In refraction, even small refractive errors can lead to visible artifacts when the ray passes through a sufficiently long distance before it hits the textured wall on the other side. More importantly, refracted rays in a small area can be deviated to quite departed area. This frequently happens for the rays which intersect with the transparent object around the fridge of the silhouette, where the normal directions have a dramatic and abrupt diversities and resulted neighboring refractive rays have really large angles among each other. These in turn cause undersampling of rendering refractive image.

Traditional super-sampling has the disadvantage of requiring an excessive amount of time, although it provides good rendering quality. Pixel supersampling [75] is a popular technique to overcome these defects, replacing them by blur and noise. However, pixel supersampling increases the rendering effort significantly, even when pyramidal rays [1] or other object-space ray aggregation or refinement constructs are employed. In our case, since we seek to gauge refractive object properties by the way they distort a simple pattern in the background, we can devise an acceleration techniques that defers the supersampling all the way to the pattern sampling stage, while preserving at good accuracy the local distortion of the (virtual) supersampling grid.

Our post-refraction supersampling scheme, on the other hand, helps to overcome irregular sampling of the background texture in a cost-effective way. It is able to save the largest part of the computational cost for the ray tracing while retaining the quality benefits of super sampling. It does not need to trace more rays, and it only performs the super-sampling on the background texture. Instead, it makes use of the space coherence of a matrix of neighboring rays. The advantage of this algorithm is that by tracking the ray's background position, we know the shape of a group of neighboring points, which gives us an imprint on to what degree the refraction has distorted the original background image for display, without the cost

of the actual supersampling. The use of post-refraction supersampling in place of the real supersampling improves performance noticeably, especially for larger or complicated datasets with multiple isosurfaces.

1.3.3 Refraction rendering in volume graphics

We present an efficient framework for the high-quality rendering of discretely sampled surface-based objects with refractive effects in volume graphics. Our framework employs a high-quality spline-based filter in conjunction with a novel filtered octree space decomposition with pre-classified cells that is carefully matched to the filter and voxel neighborhood characteristics. These leads us to achieve an accurate estimation of the refraction coefficients, an efficient and accurate surface detection, space traversal, and backdrop image sampling, while keep the system in the interactive speed.

In the early work of rendering of refractive objects [34] [75], objects were either defined as a polygonal mesh or spline patches. Currently, it is still popular in renders, such as POV-Ray [29] and others. There are much less research going on the modeling of refraction in discretely sampled objects, such as in volume graphics. In recent years, Rodgman and Chen published a serious work on this topic and indicated important issues in refraction in discrete datasets. They include the detection of the exact location of the refractive interface, the amount of noise that may exist in the volume datasets, and the interpolation filter used to estimate the density of the materials on both sides of the refractive interface.

We employ high quality filters with accurate smoothing capability for interpolation and gradient (refractive index) calculation. This is very important for transparent object rendering. Violating any of the above constraints, and especially the first, will cause rays to stray from the correct path, where even a small angular error can cause a ray to terminate at a location on the backdrop (for example, the popular checkerboard) vastly remote from the physically correct location. These errors are amplified by the length of the distance the ray traverses past the refraction location until it hits the backdrop image.

We strive to present a framework that allows users to design refractive discretely sampled objects in a near-interactive manner and with a good estimation

of the refractive effects. Considering that objects represented as a set of sample points, arranged in a three-dimensional regular grid, in volume graphics, we have found that once refractive objects are sufficiently complex in shape, or even more so, if a few refractive objects are embedded into one another, it is very difficult to predict the visual effects that result from their illumination without actually looking at the real object or a computer simulation of it. To accelerate the space traversal of ray tracing, We combines DDA stepping and octree space decomposition. To accurately and quickly search for isosurface in volume, we classify the voxel cells that may contain the iso-surface in a pre-process, either through a gradient morphologic method [66] or, better, by a novel scheme that exploits the monotonic decay, by using non-negativity properties of the B-Spline filter. Our ACCC approach for cell classification computes a cell mask that tightly fringes the refracting iso-surface, and consequently requires the testing of much fewer grid cells for iso-surface membership in the raycasting than previous strategies. With the advantage of a variety of methods, our framework allows high-quality interpolation mechanisms to be used without incurring the speed penalties that usually come with these.

1.3.4 Accurate and fast system named as Geometry Field

We proposed a novel concept, geometry field, to deal with rendering effect, such as reflection and refraction, with high accuracy in real time for complex scene with objects modeled with polygonal meshes in traditional computer graphics. Real time rendering with reflection and refraction has always drawn great attention of researchers, but it still remains a challenge, due to the expensive global computational process, especially for the intersection testing, which is not suitable for graphics hardware. We developed a real-time system for accurate reflection and refraction using ray tracing with geometry fields, which combine light fields with geometry images. A geometry field is a 4D function, which takes a ray as the input and outputs the intersection point between the ray and the surface, represented as the uv coordinates on the geometry image. It makes intersection tests much more efficient by looking-up within a fragment shader and allows other object textures to be applied, including normal map, to capture geometric subtleties. This method can be generalized to handle refraction and self-reflection. Most real-time reflection

methods make assumptions of the geometries of the reflectors or their geometric relations to the scene. In contrast, the geometry field handles arbitrary surfaces and scenes without making any assumptions.

Ray tracing is the most well known algorithm for achieving accurate renderings. Great efforts have been spent to accelerate ray-tracing by designing special hardware [77] [11, 53, 74]. Environment mapping [7, 21] is an alternative approach that provides fast reflection and refraction of the surroundings assuming the scene elements are infinitely far away from the reflector, so in turn with the difficulties to deal with objects reflecting in near surfaces.

The light field [20, 38] is one of the central concepts in image based rendering, which maps a ray to the color of the intersection point. It maps rays to the color of intersection points, which moves a large amount of computational expense to preprocessing time and therefore achieves interactive speed. Problem for light field is the lack of accuracy, especially for high-level reflection.

Our geometry field system makes full use of the combination of advantages of light fields and ray tracing, by mapping rays to the package of position, normal and texture of intersection points and converts intersection testing in traditional ray tracing to quick table lookup. It responds to the trend of fast increases in memory capacities. Besides, we combine the powerful 2D representation power of geometry image to make the lookup table more efficient and compact. Hence, the computational power of graphics hardware can be fully utilized. Without sophisticated data structures and optimizations, our algorithm has demonstrated the capability of real time rendering of inter-reflections and refraction with convincing results using current graphics hardware.

In more detail, the geometry field of a surface S is a map, where the input is an incident ray, and the output is the texture coordinates of the first intersection point. Suppose S is represented as a geometry image $\mathbf{r}(u,v)$, γ is a ray in $\mathbb{R}^3$ intersecting S. The first intersection point p is $\mathbf{r}(u_p, v_p)$. Then the map $G : \gamma \rightarrow (u_p, v_p)$ is the geometry field of surface S. From (u_p, v_p), the position and the normal of p can be obtained from the geometry image and the normal map.

The geometry field representation has several advantages. First, it effectively reduces the memory cost. In general, a geometry field is 4 dimensional. If the

position, normal and color material information are stored for each entry, the storage requirement will be extremely high. Instead, only the texture coordinates are stored, such that the size of a geometry field is within the memory capacity of current hardware. Second, due to the regular structure of a geometry field, it can be represented as a generalized texture. The whole rendering algorithm can be implemented in common graphics hardware. Third, the time cost is independent of the geometric complexity of the scene, and only dependent on the size of the geometry field. Figure 23 illustrates the concept of the geometry field.

1.3.5 Refraction correction in ultrasound CT(UCT)

We apply the wave-based ray propagation models into ultrasound computed tomography(UCT) to correct the distortion caused by refraction phenomena. Our innovative framework is inspired from the trajectories determination in images with fast wave propagation computational techniques. By modeling the wave front movement governed by Eikonal equation with fast marching method(FMM), the accurate ray direction for an arbitrary point can be derived by searching for the minimum path in the wave field between the point and the emitter. Embedding the FMM into traditional reconstruction algorithm, SART, we demonstrated that our frame work is a promising one for the efficient and accurate modeling of the propagation of acoustic waves in a refractive media.

The area of ultrasound imaging has been devoted significant work from researchers in medical school and computer science, because of its benign, noninvasive characteristics. The developed system and algorithms not only achieve impressive speed for various kind of medical problems, but also provide a routine diagnostic tool allowing it to compete with or applying with other tomography modalities. Nevertheless, producing pleasingly clear and accurate image is still challenging with ultrasound data. Especially, using ultrasound data of the organs that have strong refraction effects, it is still difficult, if not impossible, to reconstruct image faithfully, and the alias introduced by bent rays may overwhelm the subtle, yet important characteristic features in the image. Examples of this type of phenomena are frequently discovered in the real world. For example, the large subcutaneous fat

layer in breast [67] [54] falls into this category, which significantly distorts acoustic ray's direction and eventually causes apparent alias or noise in result images. The specific difficulty of curved-ray in ultrasound image originates from the strong refraction effect when acoustic rays pass interfaces between media with different speed, according to the Snell's law for refraction. Diffraction is the other possible issue to cause acoustic rays abbreviation. However, diffraction tomography is often based on weak scattering assumption [50], which is violated by strongly refracting fat layers in breast. We focus on correcting the artifacts stemming from refraction. Previous work has either not modeled bent rays at all or has inadequately eliminated bent ray distortion effects and failed to faithfully reproduce tissue properties in UCT. Furthermore, prior methods have been computationally expensive, limiting their extendibility to three dimensions.

With the introduction of curved ray propagation models into ultrasound imaging, our framework is able to improve both the velocity image reconstruction from time-of-flight data and energy image reconstruction for attenuation data, taking into account the refraction. We treat the processing of ultrasound emitting from one sender to all receivers as a processing of wave propagation, tracing the movement of the wave front using fast marching method with eikonal equation. With the fast marching method, wave arriving time for each grid point is extracted and accurate ray directions for arbitrary point is derived by searching the minimum path in the time field between the point to sender. It provides a uniform solution in finding the speed trajectory that minimizes the integral of the propagation cost function. Benefiting from its stability, during the last ten years, fast marching method has aroused from many other application domains, such as computer graphics, computer vision and semiconductor fabrication. For example, it is used in interface evolution with the fluid simulation, the shape detection and recognitions in computer vision, and industrial simulations of etching and deposition processes in semiconductor manufacturing. To our best knowledge, we are the first ones to apply it into the acoustical ultrasound image to model the physically realistic ultrasound propagation directions.

1.3.6 GPU Accelerated UCT Made Computational Practical

Transmission Ultrasound Computed Tomography (CT) is strongly affected by the acoustic refraction properties of the imaged tissue, and proper modeling and correction of these effects is crucial to achieving high-quality image reconstructions. A method that can account for these refractive effects solves the governing Eikonal equation within an iterative reconstruction framework, using a wave-front tracking approach. Excellent results can be obtained, but at considerable computational expense. It is far more efficient than a full wave equation simulation, we find that it is still not efficient enough to warrant deployment in a clinical setting. Thus, we investigate the use of commodity high-performance hardware for parallel program execution, such as multi-core CPUs, programmable commodity graphics hardware (GPU), and processor clusters. However, the inherently sequential mechanism of the FMM method poses certain trade-offs with regards to the architecture used (and also cost). This thesis works on the acceleration of three Eikonal solvers (Fast Marching Method (FMM), Fast Sweeping Method (FSM), Fast Iterative Method (FIM)) on three computational platforms (commodity graphics hardware (GPUs), multi-core and cluster CPUs), within this refractive Transmission Ultrasound CT framework. Our efforts provide insight into the capabilities of the various architectures for acoustic wave-front tracking, and they also yield a framework that meets the interactive demands of clinical practice, without a loss in reconstruction quality. These considerations and their results are presented in this thesis, with the overall conclusion being that UCT based on fast wavefront tracking has excellent potential for real life clinical deployment.

1.4 Thesis Outline

The remainder of the dissertation is organized as follows. In chapter 2 we briefly review the background of signal processing and previous work for refraction in the area of volume graphics, traditional graphics, computational tomography. In chapter 3, we introduce our work in the normal estimation especially for refraction effect with comparing BSpline family filters with several most popular filters in visualization. In chapter 4, we show our framework for high-quality refraction

rendering system for volumetric objects, with a combination of our knowledge of filters, our proposed post-refract suppersampling and the carefully designed acceleration methods for quickly locating the isosurface position. In chapter 5, we describe the concept and system of geometry field, which is a hardware accelerated real-time rendering system for reflection and refraction of complex scene with mesh objects. In chapter 6, we present our ultrasound computational tomography(UCT) framework with the capability of refraction correction. In chapter 7, we improve the computational efficiency of UCT to medical clinical requirement with the acceleration of graphics processing unit. Finally we conclude this dissertation and outline future research work in chapter 8.

Chapter 2

Background

2.1 Signal Processing and Filters

Much work has been done on filter design in volume rendering. Moller et al. [47] developed a design approach based on a Taylor-series expansion which, apart from interpolation, also considers derivative estimation, while Bentum et al. [5] employs the frequency domain to design these filters. Unser et al. [71, 72] promote the use of B-Splines for the interpolation of discrete data and give a recursive method to estimate, from the native volume data, the coefficients that comprise the volume data used for interpolation. Li and Mueller [39] propose a frequency-space approach for resampling, using the FFTW system for fast Fourier transforms.

The importance of the accuracy of gradient estimation, over that of the interpolation process, has been reported in various research papers [5, 47]. There it was found that gradients estimated with a filter based on a cubic polynomial, such as the Catmull-Rom spline, achieve satisfactory shading results. While larger, higher-order filters estimate even more accurate derivatives [47], they tend to incur excessive computational expense in the rendering process. For these reasons, the Catmull-Rom spline is generally perceived a good compromise between computational effort and accuracy, especially when high-quality rendering is the goal. On the other hand, first-order filters, as embodied by the trilinear function, are used when rendering speed is the main focus. Its simplicity also enables a general implementation in graphics hardware. The corresponding linear derivative filter is

popular for the same reasons. Finally, the nonnegativity of trilinear filters also allows an efficient analytic root finding mechanism to determine the exact location of an iso-surface [51].

2.2 Volume Graphics

Refraction has been the subject of much interest even early on in computer graphics, where Kay [34] was the first to apply Snell's law to model this optical effect. Soon after, Whitted [75] produced the now famous image of floating transparent balls over a checkerboard to show off his recursive ray tracing framework. Amanatides [1] developed the concept of cone tracing, where slender cone rays extend from each pixel, with an initial diameter the size of the pixel. These cone rays are then distorted at refractive and reflective surfaces. As mentioned above, Rodgman and Chen [56], in their pioneering work, discussed the subject of refraction in volumetric datasets at great detail. They use a discrete ray tracing approach, and in that context they discuss three measures designed to provide the material densities on both sides of a material interface required by Snell's law in order to compute the refraction angle. First, in order to locate a material interface itself, they use a small ray step size, much less than unity, Then, they describe four methods designed to ensure proper positions to sample these two materials, and they mention the difficulties associated with correctly separating two subdomains to estimate the refraction index. In another paper [57], the same authors employed regularized anisotropic nonlinear diffusion to quantify distortion and improve image quality. Their method bears some similarity to the non-linear raytracing approach described by Grller et al. [22], which illustrates the effects of continuous dynamically and chaotic fields by visualizing the way in which they bend the light rays traversing it. This approach was later extended by Weiskopf et al. [74] using a GPU-based implementation to compute the refraction of rays subject to a space-filling analytical medium, such as the relativistic field of a black hole. There, the step size of the rays is adaptively adjusted to conform to their local curvature.

2.3 Computer Graphics

Researchers have spent great efforts to accelerate ray-tracing rendering by designing special hardware [77]. With the increase of computational power and programmability, ray tracing has been implemented on GPUs [11, 53, 74]. More recently, acceleration data structures such as Kd-tree in [19] are also sophisticatedly implemented on GPUs.

Environment mapping is introduced in [7, 21] to simulate reflection of the scene, where the enviroment is infinitely distant from the object. The rendering result is inaccurate if the reflected object is close or in motion. Szirmay-Kalos et al. have presented a method in [62] to localize the enviroment map to obtain approximated ray hits on the objects in the scene.

In image-based modeling and rendering, light field and lumigraph are introduced in [20,38] to represent the flow of light through unobstructed space in a static scene. Reflections have been rendered using light field [81], layered light field and layered depth images [42], or surface light fields [76]. Hendrich et al. [28] used two light fields to simulate accurate reflections, and Masselus et al. [45] uses precaptured light fields to relight objects. Light field mapping [12] models the outgoing radiance function as a light field to allow object motion.

Geometry image was first introduce in [23], which captures geometry as a simple 2D array of quantized points. Surface signals like normals and colors are stored in similar 2D arrays. Geometry images are generalized to multi-chart geometry images [58], smooth geometry images [43].

2.4 Computed Tomography and Ultrasound

Starting from the late 70s and early 80s, experimental work in UCT has been driven by the need for real-time data acquisition and display. While recent work by Duric [18] shows promise and discusses the effect, that a scatter field with reflection and refraction properties has, the reconstruction algorithms discussed there are still limited to the straight-ray assumption.

To solve the problems associated with bent rays, Meyer [46] proposed a

method to correct for the multi-path errors using a parametric multi-path modeling and estimation scheme, while Pan and Liu [49] proposed methods to correct for refractive errors by scanning a small area around the straight line-of-sight and then using the maximum, sum or average of the area to measure the attenuation. Several researchers [3] [48] explored the use of raytracing, via ray-linking, in an iterative reconstruction framework to improve the UCT image quality. Andersen [3] proposed a ray rebinning method to generate new projection data, while Denis [14] compared several methods for ray-tracing, showing that substantial improvement over straight ray methods can be achieved for moderately refracting fields. There still remains a need for further improvement, especially in terms of computational speed and accuracy.

2.5 GPU acceleration

Refraction in computer graphics has been a subject related to ray tracing [75] since the 1980's. Ray tracing is well known for excellent quality but suffers from excessive computation. Increasing computational power through parallelism in a graphics processing units(GPU) allows interactive speed with ray or path tracing approximation mechanisms [73]. GPU uses rasterization to generate images and some basic refraction [69] [59] has been implemented using hardware-assisted approximation.

Kruger [37] proposed a reflection and refraction algorithm on GPU by rasterizing photon paths into a texture in multi-passes. Szirmay-Kalos [63] approximate ray tracing on a GPU by creating a distance impostor around a refractive object and a iterative algorithm to approximate the distance of a ray traveling in an object. Diefenbach and Badler [15] uses a GPU multi-pass algorithm to handle planar reflection, refraction and shadow. Guy and Soler [25] rendered simple convex and faceted objects with a plane-based refraction. They rendered gemstones with complex lighting behavior by calculating refracted vertices and updating the facet tree each frame using shaders. Hakura and Snyder [26] combine environment mapping for distant reflection and ray tracing for refraction and reflection in close-ups. Hu and Qin [30] present a algorithm to use binary search to determine ray-object intersections with a depth map. Wyman [78] uses a two-pass image-space approach

to approximate two-interface refractions on GPU. His following work [79] added another additional pass to render nearby geometry to texture. Recently, Davis and Wyman [13] present an algorithm to approximate refraction using a ray-depth map intersection technique.

Chapter 3

Filter

Based on the finding that refraction imposes significantly higher demands onto gradient filters than illumination and shading, we evaluate the family of spline filters as a good alternative to the cubic filters, which so far have served as the gold standard of efficient yet high-quality interpolation filters in present visualization applications. Using a regular background texture to visualize the refractive properties of the volumetric object, we also describe an efficient scheme to achieve the effects of supersampling without incurring any extra raycasting overhead. Our results indicate that splines can be superior to the Catmull-Rom filter, with potentially less computational overhead, also offering a convenient means to adjust the extent of lowpassing and smoothing.

3.1 Introduction

An integral component in the visualization of discrete data are the filters used for interpolation and derivative estimation. While the interpolation filter determines the geometric accuracy of the object estimated from the sampled data, the derivative filter affects its perceived accuracy. The latter is due to the derivative's involvement in the shading computation, which appears in its unit-length form as the normal vector. Inaccurate normals will result in the depiction of false structures and patterns on the rendered object surface. While the modulation of the normal vectors was exploited to generate fictitious patterns in bump mapping [6], this is clearly not

intended in a typical visualization where one desires to illustrate the truth. In more recent research, the derivatives have also been used to estimate the geometry itself, by determining the exact position of an isosurface via locating extreme points in the first derivative and zero-crossings in the second derivative [35] [36].

The importance of the accuracy of gradient estimation, over that of the interpolation process, has been reported in various research papers [5] [47]. There it was found that gradients estimated with a filter based on a cubic polynomial, such as the Catmull-Rom spline, achieve satisfactory shading results. While larger, higher-order filters estimate even more accurate derivatives [47], they tend to incur excessive computational expense in the rendering process. For these reasons, the Catmull-Rom spline is generally perceived a good compromise between computational effort and accuracy, especially when high-quality rendering is the goal. On the other hand, first-order filters, as embodied by the trilinear function, are used when rendering speed is the main focus. Its simplicity also enables a general implementation in graphics hardware. The corresponding linear derivative filter is popular for the same reasons. Finally, the non-negativity of trilinear filters also allows an efficient analytic root finding mechanism to determine the exact location of an iso-surface [51].

The Catmull-Rom spline, a member of the family of cubic convolution filters, is the unofficial standard for high-quality rendering in visualization. This is well justified if one is only interested in illumination and shading effects. However, our research indicates that Catmull-Rom spline filters are insufficient for high-quality rendering of refractive effects since here the error is multiplied by the length of the redirected ray before it hits an opaque surface. For example, while the refraction of a ray within a thin lens is less sensitive to the quality of the gradient estimation, due to its small path length in the refractive medium, these adverse effects are much more dramatic when the refracted ray passes through a full sphere.

For this reason, we have given renewed attention to the topic of derivative estimation, but with a second goal to preserve computational efficiency. In our study, the visualization of an object's refractive properties was the main goal, since these cannot be visualized by shading alone. While curvature can be depicted using non-photorealistic techniques [31], these techniques do not apply to depict refractive properties in the general case. Instead, we chose to visualize an object's refractive

properties by the way in which they distort the image of a known regular pattern placed on the other side, such as a flat plane textured with a checkerboard.

While the Marschner-Lobb (ML) function is an excellent means to assess and visually convey the quality of filters, it is nevertheless a rather artificial function in the sense that even dramatic error images obtained with the ML function do not necessarily lead to poor results in practical shading scenarios. Refraction, on the other hand, is a much more demanding task in that regard, exposing even small filter infidelities at no mercy. For this reason, we conducted our filter study using refraction to highlight the effects of various gradient filters. To that end, we find that filters based on quadratic and cubic B-splines, which, so far, have received little attention in the domain of visualization research, show superior results to Catmull-Rom spline filters, with the added benefits of smaller or equal computational cost. A secondary effect is that they allow a means to control the amount of data smoothing before the densities and gradients are estimated. This is done by prior computation of a coefficient volume, which, however, only has to be done once, unless the volume densities themselves are modified.

But even small refractive errors can lead to visible artifacts when the ray passes through a sufficiently long distance before it hits the textured wall on the other side. Pixel supersampling [75] is a popular technique to overcome these defects, replacing them by blur and noise. However, pixel supersampling increases the rendering effort significantly, even when pyramidal rays [1] or other object-space ray aggregation or refinement constructs are employed. In our case, since we seek to gauge refractive object properties by the way they distort a simple pattern in the background, we can devise an acceleration techniques that defers the supersampling all the way to the pattern sampling stage, while preserving at good accuracy the local distortion of the (virtual)supersampling grid. This chapter is structured as follows. First, section 2 presents theory related to splines and their properties. In Section 3, we discuss the different aspects of our approach and its study. Finally, Section 4 reports on results, and Section 5 ends with conclusions.

3.2 Spline Filters

3.2.1 General background on filters

Discrete datasets represent scalar data on grids and interpolation filters must be employed to obtain samples at off-grid positions or to calculate gradients there. The simplest filter for gradient estimation is the central difference filter. The main advantage of this filter is its computational simplicity, but it may produce inferior reconstruction quality. The Catmull-Rom spline is more complex and is commonly used as a high-quality, yet sufficiently efficient, interpolation filter. However, as we shall see, it still fails to generate satisfactory results when employed in applications with extremely high requirements, such as refraction, in which a highly accurate gradient filters is needed to determine the exact direction of refracted rays.

To interpolate a value $f(x)$ at some arbitrary coordinate $x \in \mathbb{R}^q$, we can use a linear combination of coefficients c_k.

$$f(x) = \sum_k c_k \Phi(x-k) \tag{1}$$

The implementation is called interpolating if c_k is always equal to the grid value f_k when evaluated at the point $k \in \mathbb{Z}^q$. On the other hand, the implementation is called noninterpolating, if c_k does not have to be equal to the grid value f_k. For interpolating filters, to satisfy the requirement of exact interpolation at the grid points, the basis function for interpolating filter must vanish for all integer arguments, except at the origin, where it must assume a unit value. Both the trilinear and the Catmull-Rom spline filter fall into this category. In the case of noninterpolating filters, we do not need to put constraints onto the basis functions, which gives us more freedom in their design. Instead, to enable exact interpolation at the grid points, the non-interpolating filters determine the coefficients in a prefiltering step. One of the most widely used non-interpolating basis function is the B-spline.

3.2.2 Non-interpolating filters

With well-behaved basis functions, prefiltering for coefficients establishes a one-to-one correspondence between f_k and the sequence of coefficients, c_k.

From these coefficients, the desired value f_k can be determined. Further, non-interpolating filter functions can be transformed to their equivalent interpolating functions, which builds a bridge between interpolating and non-interpolating functions in the same domain. For the multi-dimensional case, we assume that the basis function Φ is separable, that is, the data can be processed in a separable fashion, row-by-row, column-by-column, etc.

3.2.2.1 Determining the coefficients

In the non-interpolating case, to ensure an exact interpolation of the sample points,

$$f_{k_0} = \sum_{k \in \mathbb{Z}^q} c_k b_{k_0 - k} \tag{2}$$

where $b_k = \Phi(k)$. Given the basis function, this constraint gives rise to a linear system of equations of unknown coefficients c_k. One can solve this problem with any efficient solver of a linear system of equations [66] [65] [70]. Another way to look at the constraint equation is to realize its equivalence to discrete convolution:

$$f_{k_0} = (c * b)_{k_0} \tag{3}$$

Therefore, the sequences of coefficients c_k can be obtained by convolution of the sequences f_k with the convolution-inverse b^{-1},

$$c_{k_0} = (b^{-1} * f)_{k_0}. \tag{4}$$

3.2.2.2 Cardinal representation

We can write interpolating functions by expressing non-interpolating basis functions in terms of their equivalent interpolating basis functions:

$$f(x) = \sum_{-\infty}^{\infty} c_k \Phi(x-k) = \sum_{-\infty}^{\infty} f_k \Phi_{int}(x-k) \tag{5}$$

As derived in [71], the equivalent interpolating basis functions are given by:

$$\Phi_{int}(x) = \sum_{-\infty}^{\infty} b_k^{-1} \Phi(x-k) \tag{6}$$

3.2.3 BSpline

The B-spline is one of the most widely used non-interpolating filters and has numerous advantages over popular interpolating filters. It derives from the well-known property that any polynomial spline can be represented as a weighted sum of shifted B-spline basis functions and is therefore characterized by the discrete sequences of its B-spline coefficients. There is a whole family of basis functions created from B-splines β^n, in order of n.

$$\beta^n(x) = \sum_{j=0}^{(n+1)} \frac{(-1)^j}{n!}(n+1^j)(x+\frac{n+1}{2}-j)^n \mu(x+\frac{n+1}{2}-j) \tag{7}$$

where $\mu(x)$ is the unit step function

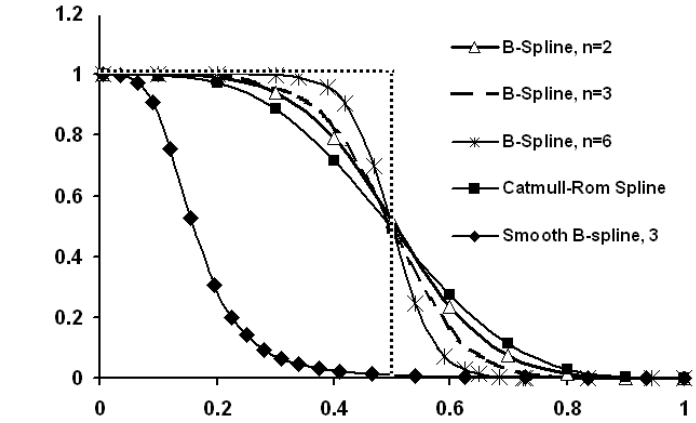
$$\mu(x) = \begin{cases} 0, if(x<0) \\ 1, if(x \geq 0) \end{cases} \tag{8}$$

The cubic B-Spline, which has a continuous second derivative, is often used in practice. Its expression is given by:

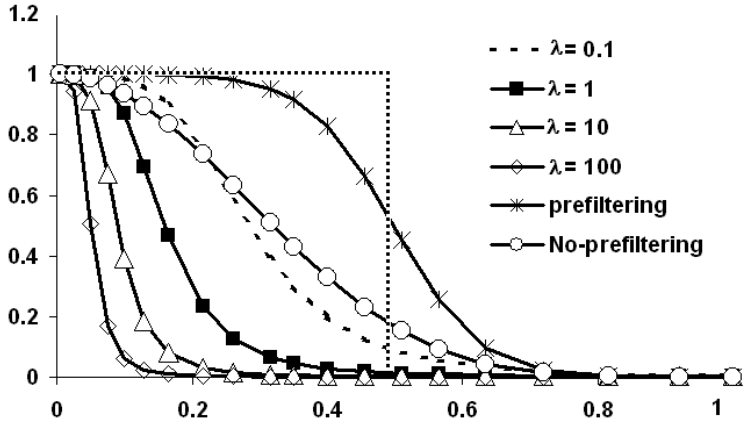
$$\beta^3(x) = \begin{cases} \frac{1}{2}|x|^3 - |x|^2 + \frac{2}{3}, if(0 \leq |x| < 1) \\ -\frac{1}{6}|x|^6 + |x|^2 - 2|x| + \frac{4}{3}, if(1 \leq |x| < 2) \\ 0, if(|x| \geq 2) \end{cases} \tag{9}$$

The cardinal representation of the cubic B-spline has a decaying oscillation that looks similar to the (optimal) sinc function (the dotted lines in Figure 1). Comparing the frequency response (see (a) in Figure 1) of the cubic convolution filter embodied by the Catmull-Rom spline and the cubic B-spline (n=3), it is clear that the cubic B-spline has a much better pass-band and stop-band behavior and is therefore superior to Catmull-Rom spline.

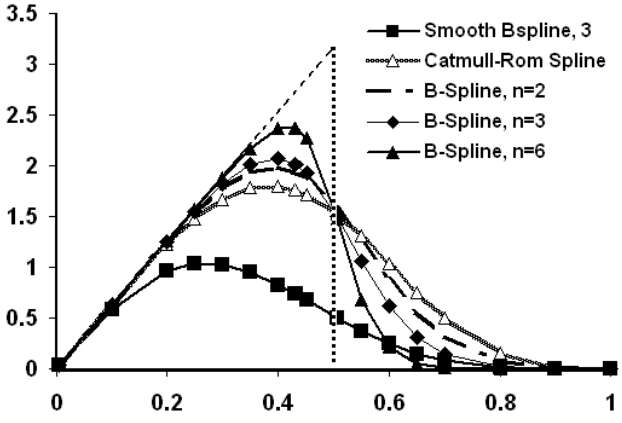
For signals that are corrupted by noise, an exact B-spline interpolation does not necessarily yield the most adequate continuous signal approximation. In this case, a more favorable approach is to employ the smoothing B-spline in place of the regular B-spline [71] [72] (see (a) in Figure 1 for the frequency response of a cubic smoothing B-Spline). The global smoothness of the interpolating function



(a) B-Spline



(b) B-Spline 3 and smooth B-Spline 3



(c) derivative B-Spline.

Figure 1: Frequency response of B-Spline related filters

can so be controlled. The coefficients are determined such that the sum of interpolating errors at the sample points and the penalty imposed by the non-smoothness of the interpolating function is minimized. A free parameter, λ,is available to control the relative importance of the two, and thus can be used to determine the amount of lowpassing. The method establishes a compromise between the desire for an approximation that is reasonably close to the data and the requirement on the function to be sufficiently smooth. The choice of λ depends on which of these two conflicting goals is given the greater importance. The same recursive technique can also here be implemented to compute the coefficients, with as few as four operations per sample value. (b) in Figure 1 shows the frequency response of a few smoothing cubic spline filters for various λ. We observe, as λ increases, the pass band is reduced and pushed towards lower frequencies, effectively smoothing the interpolation. The curves denoted Prefiltering and No-prefiltering in (b) in Figure 1 denote the frequency responses of the cubic B-spline filter with and without prefiltering the data, respectively (i.e., with and without replacing the original data points by the coefficients computed by Equation 4). Not surprisingly, the pass band for the filter without prefiltering is also reduced to a level between the smoothing spline with λ=0.1 and λ=1. This explains why using a cubic spline without prefiltering smooths or blurs the data, which has been reported by a number of authors. In fact, the prefiltering is necessary to reach the sharp transition at the Nyquist rate in the frequency response.

3.3 Application of Refraction

Refraction is the bending of a light ray as it passes across the boundary of two media. Snell's law is the most commonly used refraction equation to represent the bending:

$$n_i \sin\theta_i = n_r \sin\theta_r \tag{10}$$

where θ_i is the angle of incidence, θ_r is the angle of refraction, n_i is the refraction index of the incident medium and n_r is the refraction index of the refractive medium.

As mentioned in the introduction, refraction poses special challenges on the visualization of discrete datasets. In the following we analyze the problems at sufficient detail and devise solutions.

3.3.1 Problem description

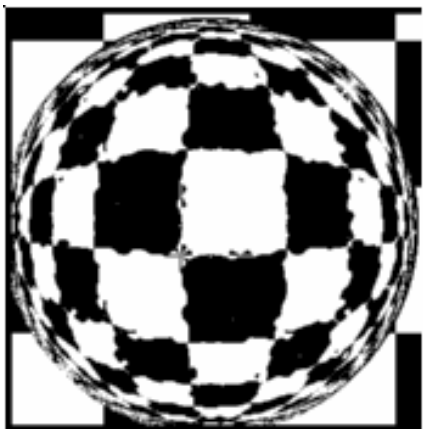


Figure 2: Poor result obtained with the cubic gradient filter, which is the derivative of the Catmull- Rom filter [5].

Figure 2 shows a refracting glass sphere in front of a checkerboard pattern. When light passes across this sphere, it bends its direction due to refraction. Although we use Brent's root-finding function [10] to determine the accurate position of the isosurface, there is still a considerable amount of aliasing in the result. There are two main reasons for this, as is explained next. First, refraction is very sensitive to gradient calculation. Even a minor error in the computed angle of gradient direction will cause the refracted light to stray far away from the correct location in the background image. The gradient is estimated by the density function in the volume, however, points in the data set are discrete-valued sampling points, which may not be able to represent the continuous density function strictly. Thus, even if an analytic function was sampled into the grid, storing the sample points in the dataset as truncated integers will result in rounding errors. On the other hand, real-world data sets, such as medical datasets, may be noisy, which will also lead to artifacts in the gradient calculation. The refraction process will accentuate any of these imperfections tremendously. Thus, a good gradient filter with just the right amount of smoothing and anti-aliasing is much desirable for high-quality refraction.Secondly, part of the aliasing stems from the sampling of the background texture at a rate lower than the Nyquist frequency requirement.

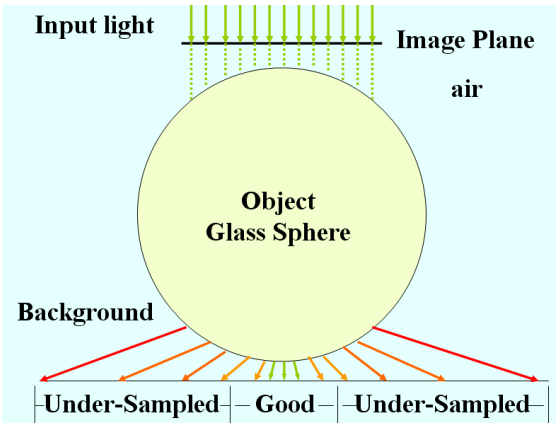


Figure 3: Refraction causes undersampling.

Figure 3 shows a one-dimensional analysis of this process. For this, we slice one plane along the x-y plane and view it from the vertical z-direction. Obviously, at the two ends of the line, the rays bent at a much larger angle than the rays in the middle of the line. This leads to the background image being sampled below the Nyquist rate at these locations. One way to overcome this problem is to use supersampling to increase the sampling rate. But this slows ray tracing considerably, since one needs to shoot more rays per pixel and compute the average.

3.3.2 Solutions for gradient computation

In refraction, the gradient filter puts special importance on the smoothness of the interpolated function. A smoothly interpolated function can ensure that gradients evaluated at two very close points on the isosurface are close in value. Therefore two parallel rays interacting with the isosurface on these two neighboring points, after refraction, can still follow near-parallel paths. This will effectively lead to a smooth image of the background texture, as seen through the refracting object. (a) in Figure 1 shows the frequency responses for the cardinal Catmull-Rom spline filter and several B-spline filters. Although the B-Spline requires a calculation of the coefficients first, this is just a one time procedure, whose outcome can be used epeatedly. We observe, based on the frequency response, that even the quadratic

B-Spline (B-Spline 2) has a better frequency response than the Catmull-Rom spline filter, although cubic Catmull-Rom spline is more expensive than B-Spline 2 (due to its wider extent), both for interpolation and for gradient calculation. With increasing order, the B-Spline becomes increasingly close to the ideal low-pass filter, the box at f=0.5. (c) in Figure 1 shows the frequency responses for the gradient (derivative) filters based on the Catmull-Rom spline and B-Spline. From these plots, we can draw the same conclusions than those been drawn from interpolation counter parts shown in (a) in Figure 1. The most interesting filter in this figure, at least for our purposes, is the smoothing B-Spline. It is a filter that fits well for the application of refraction, since here we require a good filter that also has tunable smoothing and anti-aliasing capabilities. We mentioned before that the B-Spline without precalculating the coefficients is usually thought of as erroneous, since it blurs the images. But we find that in the case of refraction, which is extremely sensitive to variations in the gradient, the smoothing effect of non-prefiltering is actually a positive aspect. (b) in Figure 1 compares the B-Spline with and without prefiltering. Globally, the B-Spline with prefiltering is much closer to the ideal low-pass filter. However, in the stop band, the B-Spline without prefiltering will suppress the aliases to the largest degree, since it smooths the image and begins to decrease earlier. This means that the B-Spline without prefiltering will curb noise, smooth the data and decrease the aliasing to a minimal level. Therefore, both the B-spline with smoothing and the B-Spline without prefiltering constitute good gradient filters for refraction calculations. The latter represents an efficient choice since it does not require the calculating of coefficients.

3.3.3 Solutions for undersampling

Supersampling is a natural solution to the problem of undersampling. It shoots a matrix of rays per pixel, possibly with jitter, and averages the result. Its downside is excessive cost. We propose a new method, named post-refraction supersampling, to save the largest part of the computational cost for the ray tracing while retaining the supersampling quality. Our method does not need to trace more rays, and it only performs the supersampling on the background texture. The key idea is to use space coherence of a group of neighboring points. Once we know the shape of

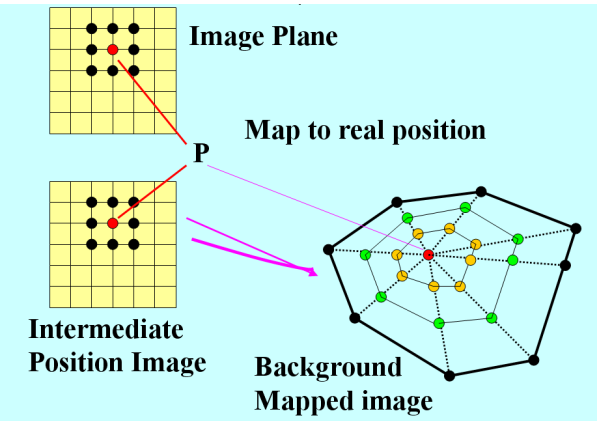


Figure 4: Post-refraction super-sampling

the refracted picture, we can do the supersamping on this known pattern. Figure 4 illustrates the steps of post-refraction supersampling. The algorithm keeps a record of each pixel's background position in an intermediate image A, which is of the same size than the original image. Then for each point p, we take the closest 8 points in A, get their positions which maps them to the background mapped image B. The number of 8 can be increased in case a higher sampling rate is needed. This tells us the refracted shape of the nine original points. This shape gives us the structure of the refraction-distorted pattern, which is what we really like to know. Finally, based on this intermediate image, we sample it by taking the intermediate points between the 8 neighbors and the center point, p. Following we do a low-pass filter convolution among the center point and the intermediate points. The result will be the supersampled density for p. The advantage of this algorithm is that by tracking the ray's background position, we know the shape of a group of neighboring points. This gives us an imprint on how far the refraction distorted the original background image, without the cost of the actual supersampling.

3.4 Experiments

To locate the exact location of the iso-surface for low order filters, such as central difference and trilinear, we use the analytical methods described in [51]. For

the Catmull-Rom Spline and B-Spline filters on the other hand, we use Brent's iterative root finding method [10] since at these higher function order no analytic root can be found. Brent's method works most efficiently for B-splines due to the non-negativity property of the B-spline basis function. The B-Spline is implemented on the method by Unser et al. [72], who gives an efficient recursive algorithm for the prefiltering of the coefficients, in Equation 4. Its computational load for the cubic B-spline only consists of two additions and multiplications per produced coefficient. Next, we will compare the central difference filter, the analytical gradient estimator based on a trilinear filter [51], and the gradient (i.e., derivative) filters [5] based on the Catmull-Rom spline and the B-Splines of order 2 to 6, respectively.

3.4.1 Gradient filters

To evaluate the different 3D gradient filters, we examined two datasets, a transparent smooth sphere with the background set to a checkerboard, and the standard filter testbed, the Marschner-Lobb dataset, described in [44]. We used a constant-valued sphere with a Gaussian fringe.

Because the Gaussian filter equation and the Marschner-Lobb function are well known, for both of the experiments, the difference between the ideal gradient and the estimated gradient by the filters can be accurately compared and investigated. In the smooth Gaussian sphere, we slice it with a plane and then choose 180 points on a circle uniformly, that is, there is one point for every two degrees. In the Marschner-Lobb experiment, we slice the dataset along a plane which is parallel to the z-axis and pick points on a circle in the same way than in the sphere experiment.

Since for both of the testbeds, the density on the chosen circle is decided only by the distance between the point and the center of the circle, the shape of the ideal x, y gradients is either a sine or a cosine curve (see a in Figure 5). In Figure 5 b-o, all such sine-shaped curves are due to the estimated gradients, while the plots immediately above each such curve show the angular error between the computed and accurate normals, and the number indicates the average error.

Since refraction requires a smoother gradient, due to the reasons discussed in Section 4.1, we also add the smooth B-Spline 3 (Figure 5f) and several B-Splines without prefiltering (Figure 5k-o). We observe, for the sphere dataset, that

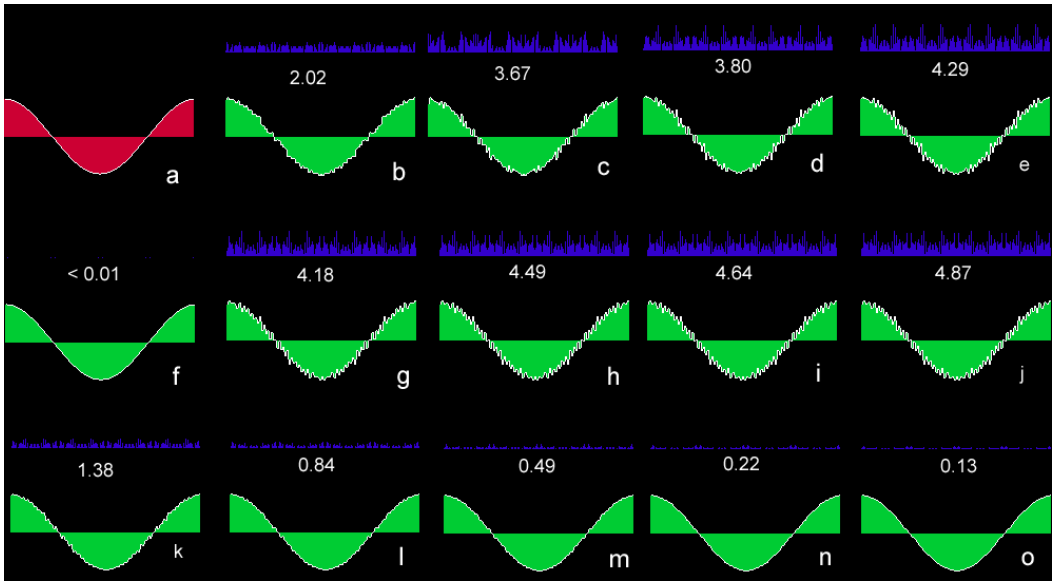


Figure 5: Gradients computed along one circle on the smooth sphere, with average error angle(up to 360). (a. perfect, b. central difference, c. trilinear-analytical, d. Catmull-Rom Spline, e. B-Spline 2, f. B-Spline3-smooth =1, g. B-Spline 3, h. BSpline 4, i. B-Spline 5, j. B-Spline 6, k-o. B-Spline without prefiltering, order 2 to 6).

the Catmull-Rom spline produces actually worse gradient results than the trilinear-analytical estimator as well as central differencing, and the B-Spline 2 to 6 (with pre-filtering) do not improve the gradient either. This is reasonable since the data have (round-off error) noise, and the better the filter the lesser is smoothed out (see Figure 1a). On the other hand, without pre-filtering, the B-Spline produces much better gradients since it has better smoothing properties (see Figure 1a), which we require in this case. The smooth-BSpline 3 offers the best results - in Figure 5f the curve of the smooth B-Spline 3 with λ=l almost matches the analytical curve, with an angular error of less than 0.01. Finally, Figure 7 presents the actual sphere refraction images, obtained with the different gradient filters. Here the smooth-B-Spline-3 shows the cleanest results, but the B-Spline without prefiltering is also able to produce a clear depiction of the distorted background.

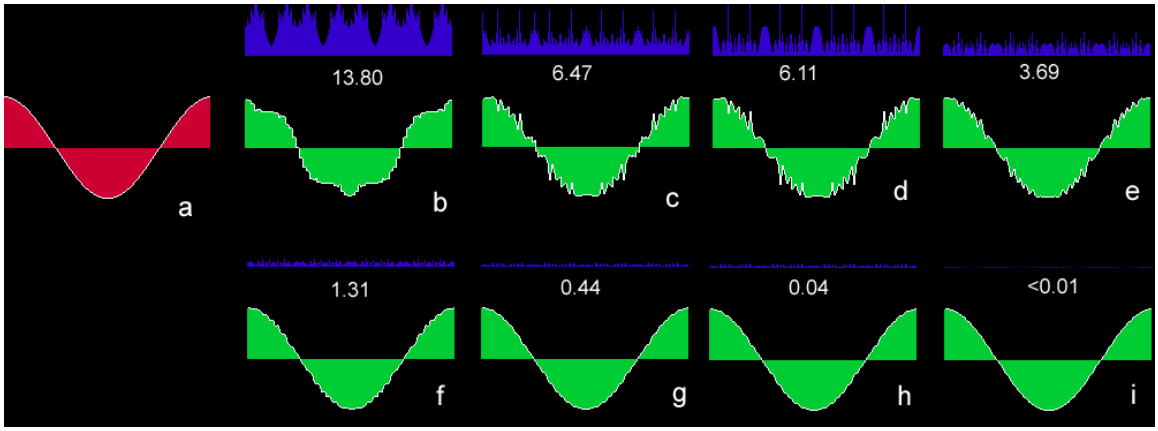


Figure 6: Gradients computed along one circle on the M-L. dataset, with average error angle. (a. perfect, b. central difference, c. trilinear-analytical, d. Catmull-Rom Spline, e. B-Spline 2, f. B-Spline 3, g. B-Spline 4, h. B-Spline 5, i. B-Spline 6)

The Marschner-Lobb test dataset, on the other hand, since it has a lot of high frequency information, requires a gradient filter that faithfully preserves the high frequency information inside the dataset. Here, no smoothing is required. As Figure 6 shows, the Catmull-Rom spline filter exhibits almost the same error than central differencing and trilinear-analytical, and the average error angle is around 6.10.

But for the B-Spline, with the order increasing from 2 to 6, the produced gradient more and more matches the ideal analytical gradient curve. The average error angle decreases from 3.69 to less than 0.01. Finally, the rendered results of the Marschner-Lobb dataset are shown in Figure 9. We observe that even the B-Spline 2 yields better renderings than the Catmull-Rom spline filter.

3.4.2 Post-refraction supersampling

Although in Figure 7, with a smooth gradient filter, the distorted background of the sphere is close to ideal, there are still some jaggies along the edges of the distorted checkerboard, especially in the sphere's periphery. Post-refraction supersampling can overcome this problem. In Figure 8, we compare the images obtained with traditional supersampling and with our fast post-refraction supersampling. The runtimes are also shown, with speedups nearly the degree of supersampling.

3.5 Conclusions

We have demonstrated, using refraction as a showcase, that the B-Spline filter achieves superior results, compared to the traditional Catmull-Rom filter, for the estimation of gradients from discrete data. It does so at the same computational cost for the B-Spline-3, which has the same support, or at 3/4 of the cost for the B-Spline-2, which has a full spatial support of 3.0. While B-Splines do require the computation of coefficients from the raw densities, to be used in the interpolation process instead, this is only required once as long as the densities are not modified, which is unlikely for pure visualization purposes. We have also demonstrated that the B-spline allows one to balance smoothing with grid sample interpolation fidelity. This is beneficial in the presence of noise, round-of errors, and other artifacts incurred in the sampling of the original data. We verified these findings using the rigorous analytical Marschner-Lobb dataset, where smoothing is not required. Our post-refraction supersampling scheme, on the other hand, helps to overcome irregular sampling of the background texture in a cost-effective way.

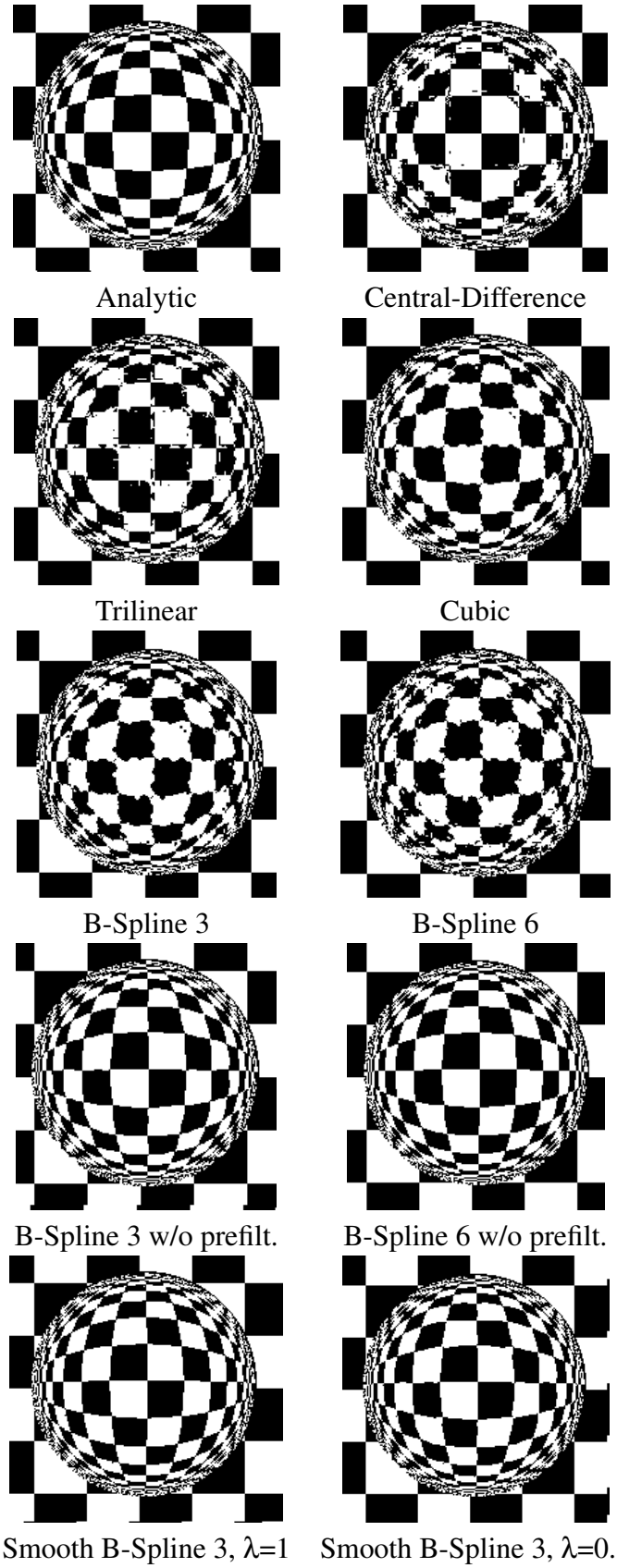
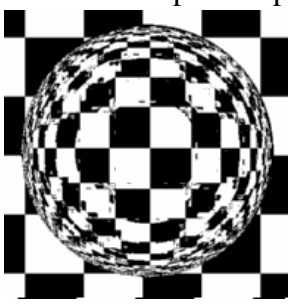
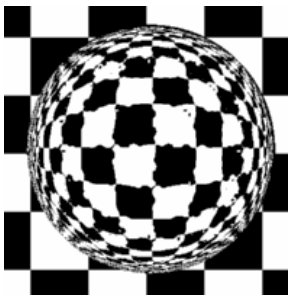


Figure 7: Refraction results of spheres with different filters.

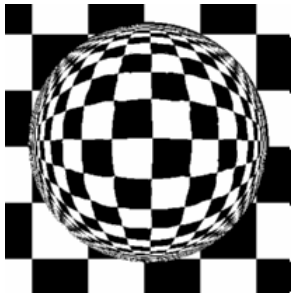
(a) Traditional super sampling



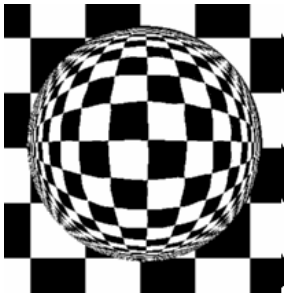
Central Difference, 9.9 ms



Catmull-Rom spline, 11.5 ms

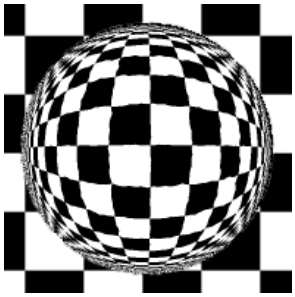


B-Spline 3, no prefiltering, 10.5 ms

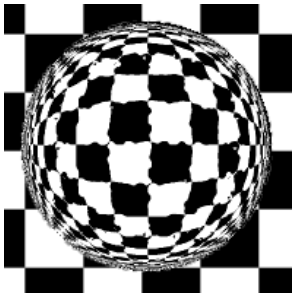


Smooth B-Spline 3, 10.5 ms

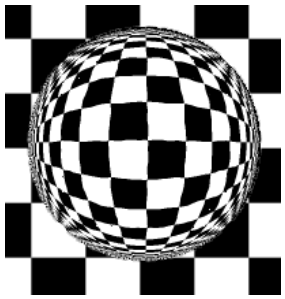
(b) Post-refraction super sampling



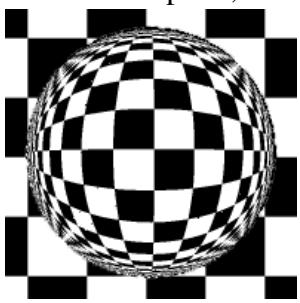
Central Difference, 2.3 ms



Catmull-Rom spline, 2.8 ms



B-Spline 3, no prefiltering, 2.6 ms



Smooth B-Spline 3, 2.6 ms

Figure 8: Compare refraction result of spheres: (a) traditional super sampling; (b) post-refraction super sampling.

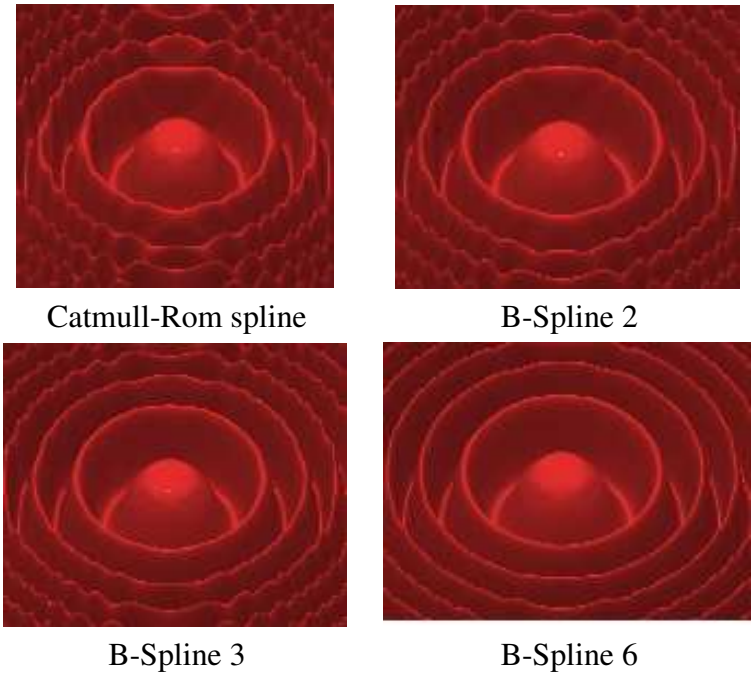


Figure 9: Rendering result of Marschner-Lobb.

Chapter 4

Fast High-Quality Refraction for Volume Graphics

We present an efficient framework for the high-quality rendering of discretely sampled surface-based objects with refractive effects. This requires an accurate estimation of the refraction indices, paired with efficient and accurate surface detection, space traversal, and backdrop image sampling. Our framework achieves these goals, by employing a high-quality spline-based filter in conjunction with a novel filtered octree space decomposition that is carefully matched to the filter characteristics. Finally, we describe an innovative scheme that achieves the high quality of pixel supersampling on a flat backdrop plane without the overhead of tracing the actual rays across the refractive object.

4.1 Introduction

The rendering of refractive objects has fascinated researchers from the early days of computer graphics [34] [75]. In that pioneering work, the objects were either defined implicitly or described as a polygonal mesh or spline patches, and this still is the representation of choice in current popular renderers, such as povray [29] and others. On the other hand, the modeling of refraction in discretely sampled objects, as constituted by volumetric datasets, has been studied much later by Rodgman and Chen [56] at great depth. In that work it became apparent that

refraction in discrete datasets is extremely sensitive to: (i) the detection of the exact location of the refractive interface, (ii) the amount of noise that may exist in the volume dataset, and (iii) the interpolation filter used to estimate the density of the materials on both sides of the refractive interface. Violating any of these constraints, but especially the first, will cause rays to stray from the correct path, where a small angular error can cause a ray to terminate at a location on the backdrop (for example, the popular checkerboard) vastly remote from the physically correct location. These errors are amplified by the length of the distance the ray traverses past the refraction location, until it hits the backdrop image. This is a prime reason why the quality of the filter used for interpolation and gradient (refractive index) calculation is so important, and the previous chapter of this thesis has addressed these filter aspects in detail. In that work, refraction was only used as a way to illustrate our new insights into filter design, since its sensitivity to filter quality represents an excellent means for a visual presentation of these quality issues. In this paper, we will only summarize these insights in order to justify some of the choices we have made.

In the work presented here, we focus on the computational aspects of refraction, in the context of volume visualization and volume graphics. More concretely, we strive to present a framework that allows users to design refractive objects, in a near-interactive manner and with a good estimation of the refractive effects. We have found that once refractive objects are sufficiently complex in shape, or even more so, if a few refractive objects are embedded into one another, it is very difficult to predict the visual effects that result from their illumination without actually looking at the real object or a computer simulation of it. In that respect, our paper is somewhat related to a recent paper on the computer modeling of gemstones [25], which employed Heckbert's beam-tracing [27] approach to manage the rays refracted at the polygonal surface that modeled the many facets of the diamond. In contrast, our paper does not target polygonally-meshed surfaces. Rather, it considers objects as being represented as a set of sample points, arranged into a three-dimensional regular grid.

This volumetric representation, when interpolated with a suitable filter, allows the modeling of arbitrarily smooth objects, where the degree of smoothness is determined by the interpolation filter and the level of available detail is determined by the

resolution of the grid. This, in fact, can be considered a strong point of a volumetric object representation and the field of volume graphics as a whole, but it poses certain demands on the filters used for object reconstruction. Another key issue is the rendering speed, as intersection calculations typically required for spline-based object rendering are now replaced by grid interpolation calculations. In this paper, we offer an elegant method based on a filtered octree to identify a refractive iso-surface quickly, while employing high-quality interpolation filters. These two meathods combined allow a near-interactive design of refractive objects with high-quality visual feedback. The structure of our paper is as follows. Section 2 addresses the theoretical aspects of this work and outlines present problems. Section 3 describes our actual implementation, and Section 4 and 5 present results and conclusions, respectively.

4.2 Issues related to refraction visualization

Refraction describes the bending of a light ray as it passes across the boundary of two media. Snell's law is the most commonly used refraction equation to represent the light's bending:

$$n_i sin(\theta_i) = n_r sin(\theta_r) \tag{11}$$

where θ_i is the angle of incidence, θ_r is the angle of refraction, n_i is the refraction index of the incident medium and n_r is the refraction index of the refractive medium (see Figure 10).

When applied in the context of volume graphics, refraction poses special challenges in the visualization of these discrete datasets, because it causes obvious aliasing artifacts in the presence of curved surfaces. In previous chapter, we analyzed these problems in detail and we also discussed the theory of proper solutions in depth. We shall now summarize these results briefly, in order to give a good idea of the problems and their solutions.

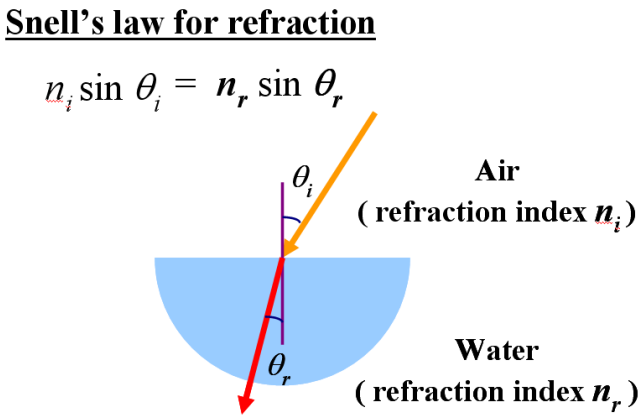


Figure 10: Snell's law for refraction

4.2.1 Gradient estimation

Refraction poses especially high demands on the accuracy of gradient estimation, since it relies on the estimation of gradients to decide the path of the refracted rays. Besides, it magnifies any gradient estimation errors by sending the refracted ray along the wrong path, which can potentially lead to more severe visual artifacts than when the normal vector is poorly estimated in illumination calculations. Also, while the gradient is estimated by the density function in the volume, points in the data set are discrete-valued, which may not be able to represent the continuous density function strictly. Thus, even if an analytic function has been sampled into the grid, storing the sample points in the datasets as truncated integers will result in rounding errors. Finally, real-world data sets, such as medical datasets, may be noisy, which will also lead to artifacts in the gradient calculation. The refraction process will accentuate any of these imperfections tremendously. Therefore, a good gradient filter with just the right amount of smoothing and anti-aliasing is strongly recommended for high-quality refraction.

In [40] we compared interpolation filters, such as the traditional cubic Catmull-Rom (or cubic convolution, CC) filter, and found that non-interpolating filters tend to give more freedom in filter design, since we do not need to put constraints onto the basis function. The B-Spline family filters, a widely used non-interpolating

basis function, estimate the function and its gradient much more accurately than CC filters. Although the B-Spline requires a pre-filtering of the data first [71], this is just a onetime procedure, whose outcome can be used repeatedly. For signals that are corrupted by noise, an exact B-spline interpolation not necessarily results in the most adequate continuous signal approximation. In this case, a more favorable approach is to employ the smoothing B-spline in place of the regular B-spline [71] [72]. Ideally, when the pass-band becomes closer to the sinc-rectangle, the image will be less smooth, while when the stop-band decreases more sharply, the aliasing will decrease. The smooth B-Spline with a free parameter, λ, is able to adjust the degree of smoothing.

4.2.2 Super-sampling

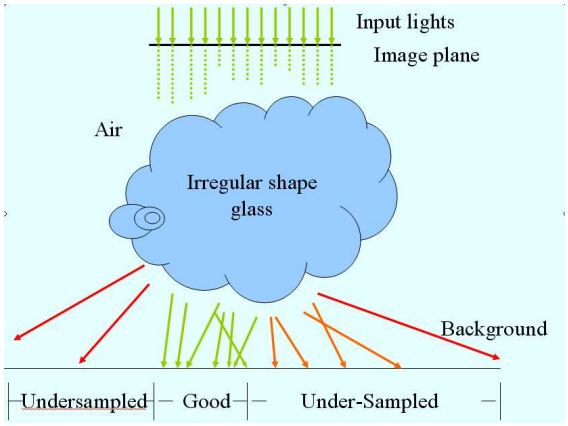


Figure 11: Illustration of object-dependent backplane sampling.

Part of the aliasing stems from sampling the background texture at a rate lower than the Nyquist frequency requirement. Figure 11 shows the effect of under-sampling, when illuminating an irregular glass sphere filled with heterogenous materials passing through by refracted light rays. Here we observe, that at some area in the background the rays bend at much larger angle than other places. This leads to the background image being sampled below the Nyquist rate at these locations.

One way to overcome this problem is to use super-sampling to increase this sampling rate, that is, send more rays per image pixel. A common practice is to shoot a matrix of rays per pixel, possibly with jitter, and then average the result. However, a major downside of super-sampling is the excessive cost incurred by having to send these many more rays across the object.

4.2.3 Refraction interface

An additional challenge comes from the need to determine where the refraction interfaces lie inside the discretely sampled datasets. We use iso-surfaces to represent the refraction interface. Here, each interpolation filter will obviously give rise to a (slightly, but crucial for refraction) different iso-surface shape, as smoothness increases with higher-order filters. To locate the exact location of the iso-surface for traditional low-order filters, such as central difference and trilinear, we use the analytical methods described in [51]. For the higher-order filters, such as cubic convolution and B-Spline, we use Brentś iterative root finding method [10] since at these higher function orders no analytic root can be found. Brentś method works efficiently for B-splines due to the nonnegativity property of the B-spline basis function.

4.3 Acceleration methods

In order to get accurate and pleasing images, refraction is usually done with raytracing, but along with this comes extensive computational cost. Although a great deal of work has been done on the acceleration of raytracing for main stream computer graphics, much less work has focused on accelerating the raytracing of sampled objects interpolated with spline-based filters. Next we describe our contributions in this regard.

4.3.1 DDA and octree

The 3D-DDA algorithm, such as Bresenhamś algorithm, is a very fast line drawing algorithm. If we combine it with the previously mentioned root-finding

methods, we can guarantee that no isosurface is missed. We use 3DDDA to step from cell to cell, and when the potential for an iso-surface exists, we perform further checks. This approach eliminates the need for high ray stepping rates (on the order of 0.1 to 0.01) of [56] and therefore provides for faster rendering.

However, the speed can still be improved by spacedecomposition techniques, such as an octree. The octree allows us to check and skip the space that does not contain an isosurface quickly. The approaches for octree traversal fall into bottom-up and top-down. Usually, bottom-up approaches start at and work entirely on leaf nodes, while top-down approaches start from the root and work down towards leaves. In order to identify large spaces with or without iso-surfaces as early as possible, we choose the top-down approach.

4.3.2 Pre-computed cell-classification

One useful property of the Spline-based filters is that the basis functions are always non-negative. This makes it possible to achieve acceleration by classifying the voxel cells that may contain the iso-surface in a pre-process, either through a gradient morphologic method [66] or, better, by a novel scheme that exploits the monotonicdecay and non-negativity properties of the B-Spline filter. Both of these methods are described next.

4.3.2.1 Binary morphologic cell-classification (BMCC)

In [66] a method using binary morphology in the preprocessing stage was proposed for an acceleration that employs traditional binary dilation and erosion operators to filter out (tag) all cells that cannot contain the iso-surface. Here, a cell is a 2^3 grid point neighborhood. This leads to a considerable reduction of cells that have to be inspected for containing the isosurface at runtime.

The B-Spline of order n is a filter with an effective support of $(n+1)$. Therefore, in 3D for a specific sample point, only a neighborhood of $(n+1)^3$ grid points will affect it. We call these points the effective supporting neighbors (see Figure 12).

In the binary morphologic cell classification (BMCC) method, after the spline coefficients have been computed (via pre-filtering), the iso-value is subtracted from these coefficients and a binary coefficient representation is generated, setting 1 for

a subtracted coefficient larger than zero and otherwise 0. Then the morphological gradient (dilation subtracting erosion) is applied to the binary coefficients and only those cells with effective supporting neighbors that have a mix of 0s and 1s in their binary coefficients are flagged with 1. These are the cells that potentially contain an isosurface since the local density field generated by their effective neighbors may transition from 0 to 1, and vice versa. Effectively, dilation expands the outside boundary of the object, and erosion removes the inside volume. After dilation subtracting erosion, a thick binary fringe that lines the iso-surface is left.

Although effective, the membership requirement of the binary morphological method is too conservative, leaving the remaining fringe too wide. This is because the method does not take into account the relative importance of neighboring points as a function of distance (which maps directly to the weighting with the interpolation filter), as well as their values, in the determination of their contribution to the interpolated value at an arbitrary sampling point x. Every neighbor which can affect the value of x has been given the same influence on x, which inflates the number of iso-surface candidates, and in turn causes a unnecessary large number of false alarms when searching for the iso-surface as a ray traverses the grid. Consider the case in which all neighboring points have a value below the iso-value, except one point that is relatively far away and of a value slightly greater than the iso-value. In that case, it is almost certain that the interpolated value of this point will be below the iso-value. However, the binary morphologic cell-classification just presented will flag this cell as a possible isosurface cell and will prompt more expensive further tests when encountered by a traversing ray. Next, we describe a new non-binary scheme that is more sensitive to these issues.

4.3.2.2 Continuous cell-classification (CCC)

Determining the maximum and minimum possible value inside a cell is an optimization problem for maximizing and minimizing the interpolating function:

$$f(x) = \sum_k c_k \Phi(x-k), (i \leq x \leq i+1), k, i \in Z^q \tag{12}$$

Here, c_k are the coefficients for each grid point while $\Phi(x)$ is the basis function. For the quadratic and cubic B-spline, this optimization problem can be solved

as follows. First, we find the set of stationary points by solving the first partial derivatives set to zero. Then for each such stationary point, we determine its eigenvalues by ways of the Hessian matrix which is based on the second partials. If the eigenvalues of a stationary point are all positive, then the point is a local minimum point, while if the eigenvalues are all negative, it is a local maximum. These maxima and minima points form critical points, and an interpolation at these will yield the minimum-maximum value range of the cell.

However, this calculation is quite time-consuming. The method needs to solve this optimization problem in each cell of a volume. Furthermore, due to the boundedness to the cell, there are several different special cases, which require additional computations. The critical point may not fall into a cell, but may appear on faces, edges or vertices, as shown in Figure 13. The optimization problem needs to be solved for each of these. Finally, for a B-spline of order larger than 3, the high order first and second partial derivatives are difficult or impossible to solve. We therefore decided to employ an approximate method, as is described next.

4.3.2.3 Approximate CCC (ACCC)

In light of these arguments, we propose a novel method we call approximate continuous cell-classification (ACCC). It is not as analytic as CCC, but it considerably tightens the iso-surface candidate requirement of BMCC, taking into account both the grid point values and the underlying B-Spline interpolation filter, but avoiding the excessive computational cost of CCC.

Figure 14 sketches the idea in 1D. For any point between grid point i and i+1, its value is always within the range Sum(MINj) and Sum(MAXj), where MAXj and MINj are the maximum and minimum values, respectively, of the voxel-scaled bases functions on grid point i and i+1, with kernel j traversing across grid cell (i, i+1). To accomplish the cell-tag volume, one can simply splat each voxel into a cell grid, weighted by its value and the basis function values at the cell boundaries (to contribute to the MIN and MAX sums, as further explained in the caption). The aggregation of all such splats will then determine if the cell is likely to participate in the iso-surface or not. It will if the value range (called the approximate range) contains the iso-value (or crosses zero if this process follows the subtraction mechanism mentioned in Section 4.2.1).

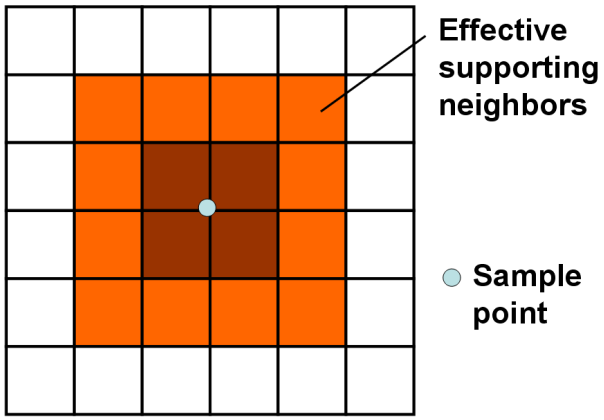
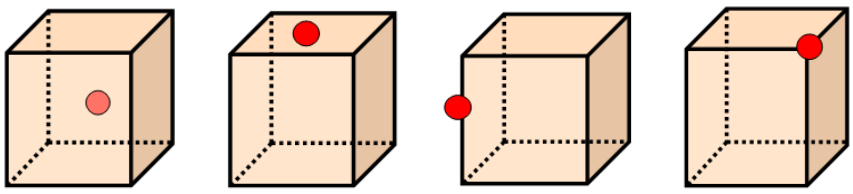
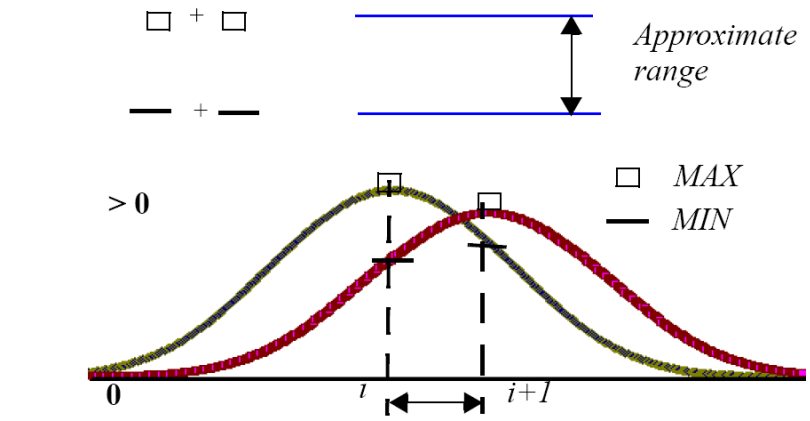


Figure 12: All colored cells contain effective supporting neighbor points for the sample point shown, here for a B-Spline of order 2. The darker the color is, the larger the support.

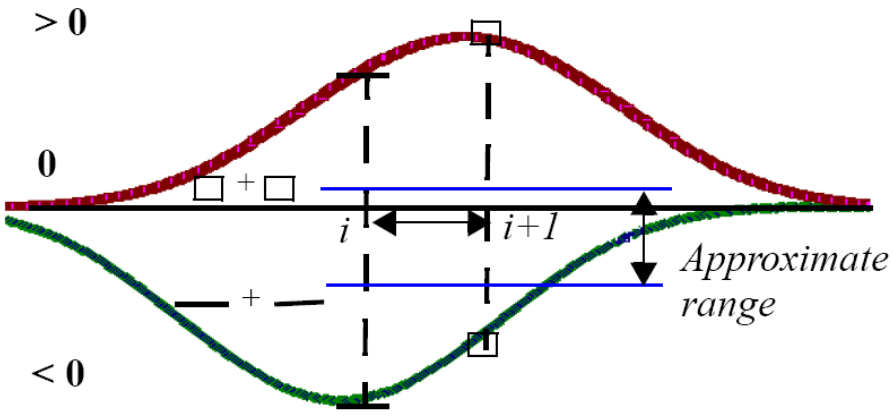


(a) inside a cell (b) on a face (c) on an edge (d) on a vertex

Figure 13: Cases for critical point with maximum or minimum value in a cell.



(a) Both grid points are positive



(b) One grid point is negative

Figure 14: Sketch of the ACCC method.

Figure 15 illustrate ACCC method with two kernels. The kernel for any sample point between grid point 3 and 4 is within the narrow band of the kernels on grid point of 3 and 4. More formally, when the absolute distance between a grid point and an arbitrary sample point x is within i and $(i+1)(i \in Z, i > 0)$, the interpolated value at x is:

$$f(x) = \sum_k c_k \Phi(x-k) = f_{iso} + \sum_k (c_k - f_{iso})\Phi(x-k) \tag{13}$$

where f_{iso} is the iso-value, ϕ is the (monotonically decreasing, non-negative) kernel function, and $k \in Z^q$.

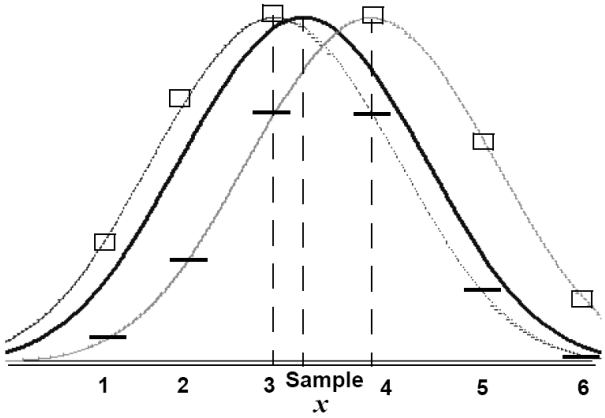


Figure 15: Illustration of the ACCC method with two grid point kernels.

Now consider x to be located between i and $i+1$. Then we know that the upper bound of the interpolation value that x can take is given by:

$$\begin{aligned} Max(f(x)) &= f_{iso} + Max(\sum_k (c_k - f_{iso})\Phi(x-k)) \\ &= f_{iso} + \sum_k (c_k - f_{iso}) \begin{cases} Max\Phi(x-k), if(c_k - f_{iso}) \geq 0 \\ Min\Phi(x-k), if(c_k - f_{iso}) < 0 \end{cases} \end{aligned} \tag{14}$$

where

$$\begin{aligned} Max\Phi(x-k) &= \Phi(i), \quad where(i \leq |x-k| < i+1), \\ Min\Phi(x-k) &= \Phi(i+1), \quad where(i \leq |x-k| < i+1), k, i \in Z^q \end{aligned} \tag{15}$$

Likewise, the lower bound of the interpolation value for x is given by:

$$\begin{aligned} Min(f(x)) &= f_{iso} + Min(\sum_k (c_k - f_{iso})\Phi(x-k)) \\ &= f_{iso} + \sum_k (c_k - f_{iso}) \begin{cases} Min\Phi(x-k), if(c_k - f_{iso}) \geq 0 \\ Max\Phi(x-k), if(c_k - f_{iso}) < 0 \end{cases} \end{aligned} \tag{16}$$

Therefore, by computing the lower and upper bound of the interpolation value that x, located between discrete point i and $i+1$, can take, we can give a good estimate whether the region between i and $i+1$ contains the iso-value or not.

This logic can be easily extended to 3-D interpolation, taking advantage of the fact that the interpolating function is separable:

$$\begin{aligned} f(x) &= \sum_{i,j,k} c_{i,j,k}\Phi(x-i)\Phi(x-j)\Phi(x-k) \\ &= \sum_{i,j,k} (c_{i,j,k} - f_{iso})\Phi(x-i)\Phi(x-j)\Phi(x-k) + f_{iso} \end{aligned} \tag{17}$$

Therefore the upper bound for the interpolation value of an arbitrary point x location, located in the cell $(i,j,k) - (i+1, j+1, k+1)$, is given by in equation 18 below. The lower bound can be derived similarly.

$$\begin{aligned} Max(f(x)) &= f_{iso} + Max(\sum_{i,j,k} (c_{ijk} - f_{iso})\Phi(x-i)\Phi(x-j)\Phi(x-k)) \\ &= f_{iso} + \sum_{i,j,k} (c_{ijk} - f_{iso}) \begin{cases} Max\Phi(x-i)Max\Phi(x-j)Max\Phi(x-k), if(c_{ijk} - f_{iso} \geq 0) \\ Min\Phi(x-i)Min\Phi(x-j)Min\Phi(x-k), if(c_{ijk} - f_{iso} < 0) \end{cases} \end{aligned} \tag{18}$$

4.3.2.4 Discussion

In [66], Thevenaz and Unser proposed a voxel pruning scheme to improve their binary morphologic method and reduce the number of voxels that are flagged as possibly containing the iso-surface. For this, they use the multiresolution space embedding property of splines. They revisit each relevant voxel and compute finer

level results. However, the relevant coefficient numbers can grow extremely large and each finer level results in a doubling of the number of coefficients along each dimension, which grows very quickly as the finer level depth increases.

Comparing ours to this voxel pruning method, we note that the cost of our method is very small. One positive aspect of our ACCC method is that we only consider the coefficients and weights of the filter at the integral grid points of the dataset. This leads to at least two advantages: First, we do not need to increase the number of coefficients – our calculations are only based on the original coefficients. Second, we do not require a calculation of the spline filter values at each grid point. Since we only need $(n+1)$ integer point weights for a spline of order n in one dimension, we can implemented this as a small lookup table of size $(n+1)$. In cubic splines, for which each voxel that was flagged as 1 in the binary gradient process, we only need to perform four multiplies and four additions in one dimension.

ACCC tightens the range of cells which possibly contain the iso-surface. For example, in a smooth sphere of size of $64x64x64$, after the tagging step the number of cells that are assigned an iso-surface membership is reduced by around 35% for the B-Spline 2 (quadratic BSpline), by 60% for the B-Spline 3 (cubic B-Spline) and by 70% for the B-Spline 6 (B-Spline of order 6).

We may make the value range of cells even tighter by dividing one cell into several sub-cells, employing the ACCC method to each sub-cells separately. We would then obtain a tighter value range within each sub-cell due to the smaller weight range for filter, and combine the value range of the whole cell from those sub-cells. The subdivision depth could be controlled by the local variation of the density field for better efficiency. Similar to the voxel pruning scheme of [66], this progressive refinement strategy would trade computational efficiency for higher accuracy.

4.3.3 Post-refraction super-sampling

Traditional super-sampling has the disadvantage of requiring an excessive amount of time, although it provides good rendering quality. Here, we propose a new method, named post-refraction super-sampling (PRSS), to save the largest part of the computational cost for the ray tracing while retaining the quality benefits of

super sampling. Our method does not need to trace more rays, and it only performs the super-sampling on the background texture. We should note at this point that our method assumes a planar background texture, and it also assumes that there are no textured objects with extensive shading effects along the ray path. With the current framework, it could be extended to handle non-planar background objects, by ways of a warping procedure. We find, however, that for testing the refractive properties of an object, a flat background object is in fact most useful since then the only distortion comes from the object under evaluation.

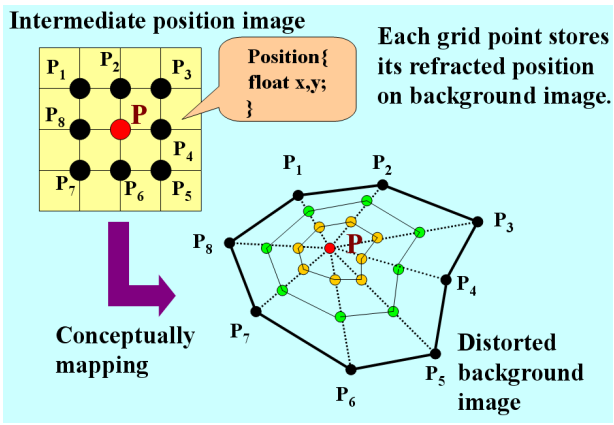


Figure 16: Post-refraction super-sampling

Our PRSS method makes use of the space coherence of a matrix of neighboring rays. Once we obtain each rays position on the background plane, we know the shape of this refracted ray matrix and the super-sampling can be done based on the known ray matrix arrangement. Figure 16 explains the individual steps of our postrefraction supersampling. The algorithm keeps a record of each pixel's background position in an intermediate image A, which is of the same size than the original image. Then for each point p, we take the closest 8 points in this image A, get their positions, which we then conceptually map to the background image. Here, the number of 8 can be increased if a higher sampling rate is needed. The resulting pattern (the distorted background image in Figure 16) tells us the structure of the refraction-distorted local image pixel grid, which is what we really like to know. Based on this pattern, we sample A and take the intermediate points between

the 8 neighbors and the center point p. Then we do a low-pass filtering along the center point and the intermediate points. The result yields the supersampled density for the center point p. The advantage of this algorithm is that by tracking the ray's background position, we know the shape of a group of neighboring points, which gives us an imprint on to what degree the refraction has distorted the original background image for display, without the cost of the actual supersampling.

The use of post-refraction supersampling in place of the real supersampling improves performance noticeably, especially for larger or complicated datasets with multiple isosurfaces.

4.4 Experiments

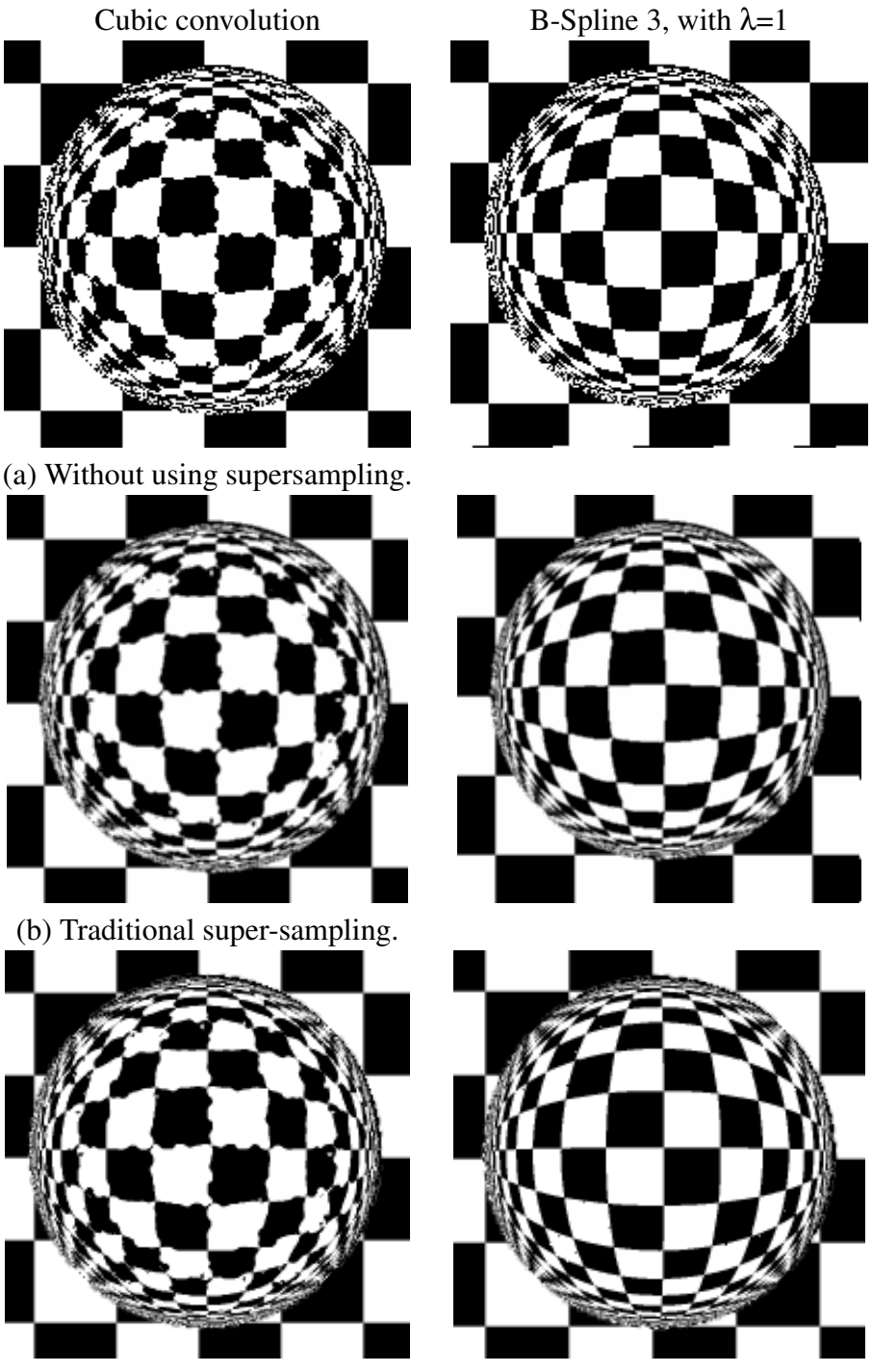
This section presents results we have obtained with the methods just described. All results were generated on a Pentium(R) processor running at 1.5GHz with 768MB of RAM. We did not use the GPU other than for display.

4.4.1 Effects of gradient filters and supersampling

The assessment of different gradient filters and sampling rate for the task of refraction was first performed using a simple transparent smooth sphere (constant-valued sphere with a Gaussian fringe, of size of 64^3) with the background set to a checkerboard.

Figure 17 shows the sphere refraction images, comparing the smooth B-spline 3 with the traditional Catmull-Rom cubic convolution filter using three different sampling methods: without supersampling, traditional supersampling, and our post-refraction supersampling. We observe that the smooth B-Spline 3 achieves much clearer pictures of the distorted background than the traditional Catmull- Rom cubic convolution, while both have nearly the same run time (See Table 1). The better image quality comes from the smoothing capability of the B-Spline, which removes much of the noise caused by integer rounding and also reduces the corresponding aliases.

The same figure compares the volume rendering results of our post-refraction super-sampling (PRSS) with those obtained with traditional super-sampling (real



(a) Without using supersampling.

(b) Traditional super-sampling.

(c) Post refraction super-sampling.

Figure 17: Refracted smooth sphere comparing Catmull-Rom Cubic and B-spline 3 with smooth λ=1 in three different categories.

SS) and without super-sampling. We observe that without supersampling some jaggies along the edges of the distorted checkerboard remain, especially in the spheres periphery. On the other hand, PRSS preserves the quality of the traditional method, but requires much less time and space. We find that the speedup factor of PRSS is nearly equal the degree of super-sampling.

The rendering experiments reported in Figure 17 and Table 1 were performed with raytracing at a small stepsize of 0.1 (to make sure that no silhouette iso-surface point is missed). These experiments indicate that PRSS incurs a time penalty of less than 2% compared to a rendering without supersampling, while traditional supersampling incurs a time penalty of over 300% compared to nonsupersampled rendering (see Table 1). More dramatic speed-ups can be expected for larger datasets. We should note, however, that the method is only an approximation and may not be stable enough for sharply changing refraction patterns in a small local neighborhood on the object. However, we have found that it has worked surprisingly well for the objects we have rendered (see below).

4.4.2 Acceleration methods

Based on the initial version of our ray tracer with PRSS and the gradient filter as B-spline 3, we applied several acceleration methods, BMCC, ACCC, DDA, and octree.

The ACCC method gives much more accurate information and decreases the number of isosurface-containing voxel candidates to around half for spline 3. This helps in the octree-guided ray acceleration to quickly detect the cells containing the iso-surface. Table 2 compares the time required for ray casting (with stepsize 0.1) to DDA and our octree methods with two cell-classification method respectively, BCC

Method	Without SS	Real SS	PR SS
C.C.	7.63	28.02	7.72
B-Spline 3	6.65	26.83	6.71

Table 1: Time (in secs) required to produce the images of Figure 17, rendered with raytracing at a stepsize of 0.1.

and ACCC, using PRSS and B-spline 3, as well as Brent method to compute the isosurface penetration exactly. On average, DDA increases the speed dramatically compared to stepped raycasting by a factor of around 7, while the octree improves matters by another factor of 3, compared to DDA. As for the cell-classification methods, ACCC saves 50% of the time-cost of BMCC. Further data for the success of the ACCC method (over BMCC) is given in Table 3. There, we can see that the higher order the filter, the more savings can be obtained (as mentioned before in Section 4.2.4).

We describe the acceleration of DDA and octree for different filters in Table 4. On average, the octree saves about 76% of the time. One interesting point is that all BSplines, even B-Spline 6, run much faster than the cubic convolution filter. This is because the latter cannot take advantage of the acceleration via our cell classification scheme since its basis function has a negative lobe, which violates the condition for this acceleration. The linear filter leads in the octree acceleration, but is closely followed by the much more accurate B-Spline filters.

4.4.3 Images obtained with refracting objects

Using the artificial smooth sphere produced from the Gaussian function, we shall now have a look at the scenario of refraction at multiple interfaces. For this, instead of just one layer of Gaussian fringe, we added one or two smooth holes into the original sphere (see Figure 18 and Figure 19). The resulting images also illustrate the complexity that can result once there are chains of refractive surfaces.

	Raycast	BMCC		ACCC	
	step=0.1	DDA	Octree	DDA	Octree
Sphere	6.65	0.84	0.29	0.69	0.15
CT-head	40.7	8.89	3.30	4.57	1.11
Lobster	83.86	14.62	6.81	8.25	2.39
Teapot	287.43	41.61	17.71	12.16	2.42

Table 2: Times (in seconds) for the acceleration of raycast with. DDA vs. Octree and BMCC vs. ACCC.

A ball with no air holes

A ball with one air hole

A ball with two air holes

A ball with different backdrops

Figure 18: Various volume graphics renderings of refractive glass balls.

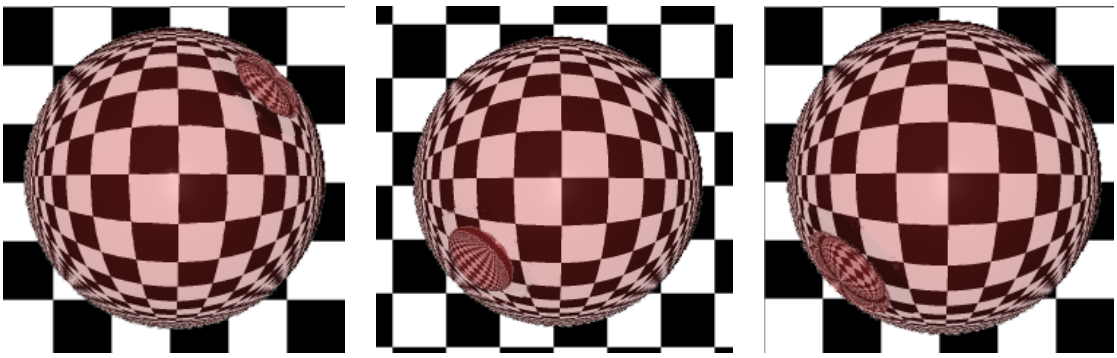


Figure 19: Multiple layers of refraction in a sphere.

For example, in Figure 19 (b) and (c), the small spheres are originally in opposite positions inside the big sphere, while the refraction image places them both in the lowerleft part of the image plane. In fact, this is reasonable since the two inner spheres are both on the route of the same input rays, considering the fact that rays are crossing inside the sphere in the diagonal direction. The difference between the inner and outer refraction layers of (b) and (c) causes different curve directions of the distorted checkerboard.

Finally, Figure 20 represents the refraction results obtained with a volumetric teapot, while Figure 21 shows a refraction image for the CT-Head. In these two cases of medical volumes, the smoothed B-Spline illustrates the clearest shape of the background, while best suppressing the noise. More refraction images are presented in Figure 22, now using a photograph as the background image to act as an

Data	Size	B-Spline 2	B-Spline 3
Sphere	64x64x64	67.1%	37.5%
CT-head	128x128x128	75.2%	52.3%
Lobster	320x320x34	62.3%	60.6%
Teapot	256x256x178	77.1%	51.4%

Table 3: Ratio of grid cells tagged to possibly containing the isosurface with the ACCC vs. BMCC method.

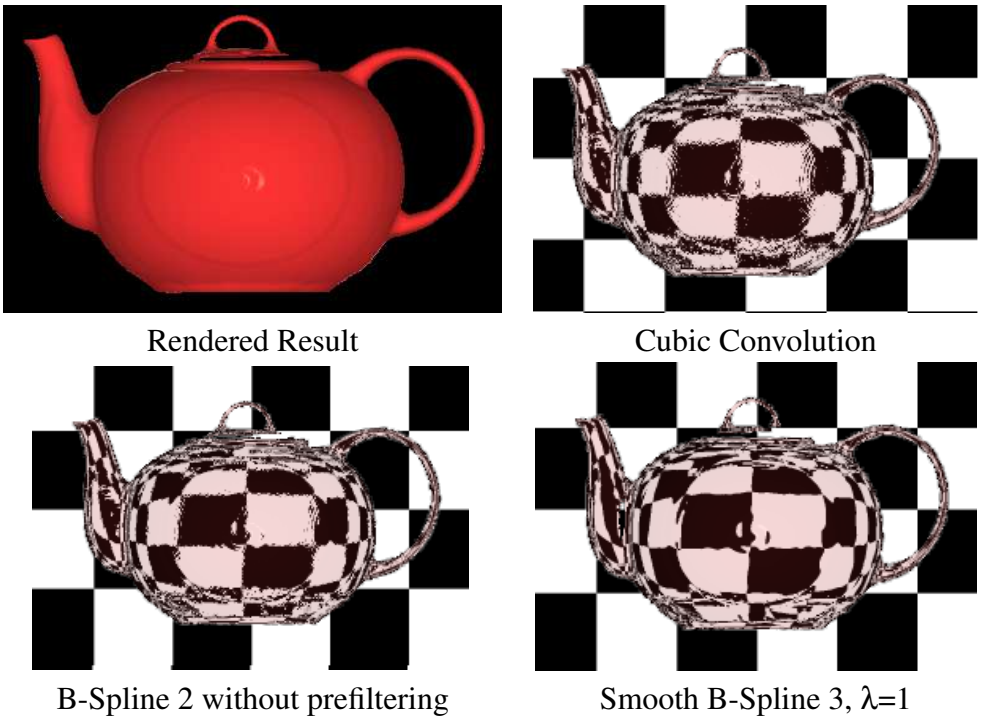
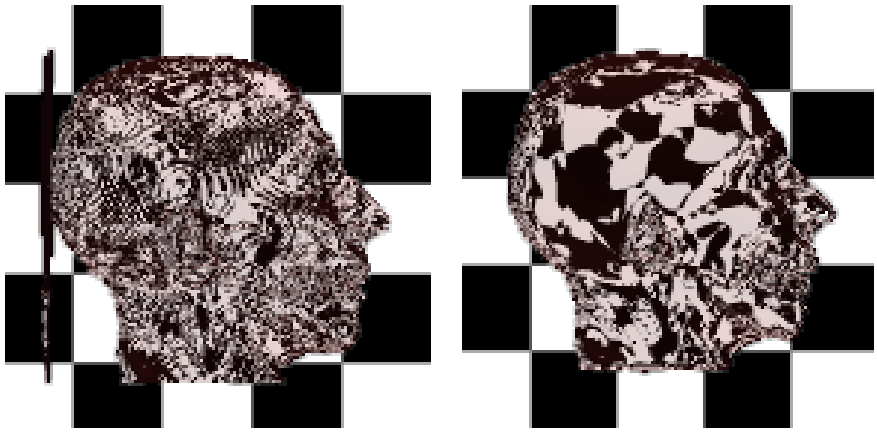


Figure 20: Refracted results of teapot.

imposter for a real scene.



Rendered Result Cubic Convolution

Figure 21: Refracted results of CT-head.

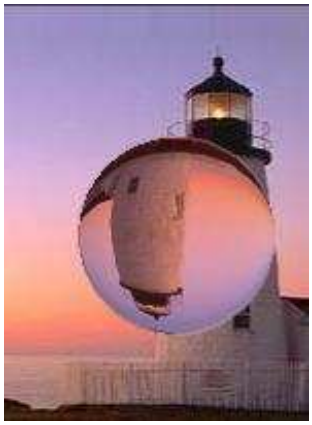
4.5 Conclusions

Obtaining high-quality refraction effects at reasonable rendering speeds has been so far a challenging task for volume graphics applications. In this paper, we have introduced a variety of methods that allow high-quality interpolation mechanisms to be used without incurring the speed penalties that usually come with these. For this, we have described a novel octree-based ray acceleration method which specifically takes advantage of the nonnegativity property of B-spline kernels. These in turn produce results superior to the ones obtained with the more

	DDA					Octree					Save
	Linear	C.C.	B-Spline 2	B-Spline 3	B-Spline 6	Linear	C.C.	B-Spline 2	B-Spline 3	B-Spline 6	
Sphere	0.93	1.79	0.67	0.69	0.89	0.09	0.42	0.14	0.15	0.34	75.2%
CT-head	5.98	7.50	3.66	4.57	10.34	0.56	2.69	1.00	1.11	1.91	76.9%
Lobster	7.44	12.46	6.33	8.25	29.10	1.48	6.49	2.32	2.39	4.07	69.7%
Teapot	30.26	34.92	8.23	12.16	21.99	1.96	5.91	2.22	2.42	3.99	82.2%

Table 4: Times (in seconds) for the acceleration with DDA and octree for different filters. On average, the octree saves about 76% time for DDA.

Teapot



Smooth sphere

Lobster

Figure 22: Refracted image results of teapot, smooth sphere, and lobster using a Smooth-B-Spline 3.

popular Catmull-Rom spline (which also does have this property). Our ACCC approach for cell classification computes a cell mask that tightly fringes the refracting iso-surface, and consequently requires the testing of much fewer grid cells for iso-surface membership in the raycasting than previous strategies. Further, we have described a heuristic, yet effective method that defers supersampling to a post-rendering process, achieving results of very similar rendering quality than traditional supersampling, for a good variety of volume graphics objects.

As future work, we seek to further explore the power of volume graphics representations to approximate continuous objects at high quality, using the B-spline filters studied in this paper. In particular, we would like to compare the directly interpolated results obtained with volume graphics methods with those generated when rendering an intermediate triangulation of the discrete object, both in terms of image quality and rendering speed.

Finally, we also believe that by porting our methods to GPUs truly interactive speeds can be reached, and this is also subject to future research.

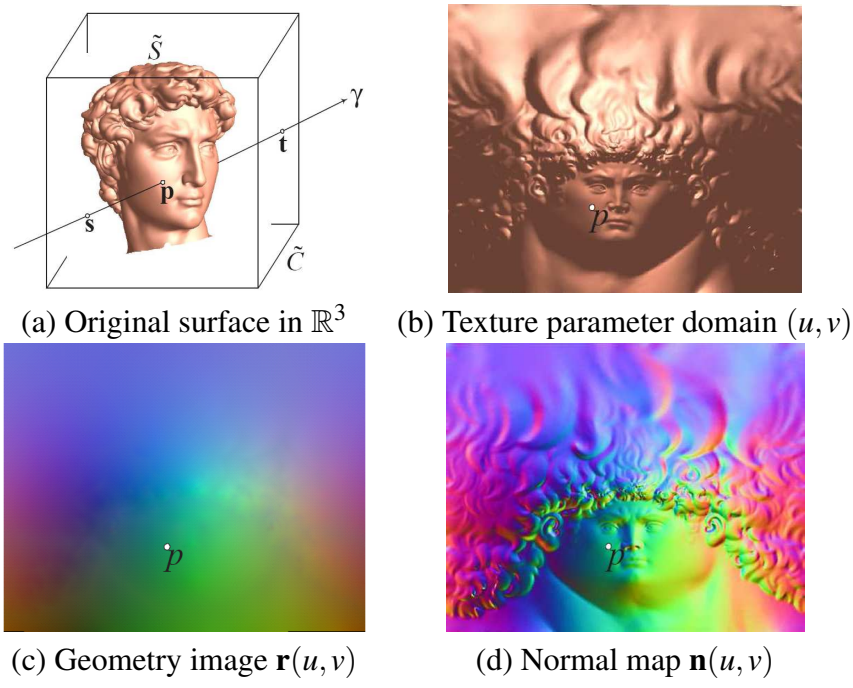
Chapter 5

Geometry Field

Geometry Field is a new method for real-time reflection and refraction using ray tracing. It combines light fields with geometry images. A geometry field effectively represents all possible ray/object intersections with geometry information, such as position, normal, material and texture coordinates, such that straightforward table look-ups replace time-consuming intersection calculations. The method is efficient, general, and easily implemented on the GPU. Our system contains multi-reflection, self-reflection and refraction. The power of this method is demonstrated by our experiments on real time reflection rendering for complex scenes using a standard GPU.

5.1 Introduction

Light reflection on mirror objects has always been of great interest in the computer graphics community. Currently developed algorithms can achieve magnificent rendering results for various kinds of materials and environments. Nevertheless, handling reflection in real time with pleasing quality remains a challenge. Furthermore photo-realistic global illumination is still very difficult if not impossible in real time due to its extraordinary complex global computational process. The specific difficulty of gaining speed in high accuracy rendering originates from the global and nonlinear computational process, such as intersection testing and visibility testing, both of which are not suitable for graphics hardware.



(a) Original surface in $\mathbb{R}^3$ (b) Texture parameter domain (u,v)

(c) Geometry image $\mathbf{r}(u,v)$ (d) Normal map $\mathbf{n}(u,v)$

Figure 23: Geometry Field.

Ray tracing is a well known and widely used algorithm for achieving accurate images. However, it suffers from slow speeds since the algorithm is not suitable for graphics hardware. Environment mapping is an alternative approach that provides fast reflection of the surroundings. However, an implicit assumption in environment mapping is that all the scene elements are located infinitely far away from the reflecting surface. When environment objects are relatively close to the reflectors, the result of environment mapping becomes inaccurate.

Inspired by the concepts of light fields and geometry images, we have developed the novel concept of *geometry fields*, which converts intersection testing to table lookup. Hence, the rapidly increasing computational power of graphics hardware can be fully utilized. Without using any carefully designed, sophisticated data structures and optimizations, our simple concept-proval algorithm has demonstrated the capbility of real time rendering of inter-reflections.

A light field is one of the central concepts in image based rendering. Suppose S is a surface in the Euclidean space $\mathbb{R}^3$. Any incident ray γ will intersect S at several points, assume the closest one to the view point is $p \in S$. Suppose the local color at p is $\mathbf{c}_p$. A light field is a map $L : \gamma \rightarrow \mathbf{c}_p$ from the incident ray to the color at the first intersection point. Light fields of general meshes can be constructed easily using conventional ray-tracing.

A geometry image represents a surface with an image. The color $\mathbf{c} = (r,g,b)$ at a pixel with texture coordinates (u,v) represents the position vector of the surface $\mathbf{r}(u,v) = (x(u,v), y(u,v), z(u,v))$. Similarly, a normal map represents the normals of a surface by an image and the color $\mathbf{c}(u,v)$ represents the normal $\mathbf{n}(u,v)$. General meshes can be converted to geometry images using parameterization methods.

A geometry field of a surface S is a map similar to a light field, where the inputs are incident rays, but the output is the texture coordinates of the first intersection point. Suppose S is represented as a geometry image $\mathbf{r}(u,v)$, a ray γ is in $\mathbb{R}^3$ intersects S. The first intersection point p is $\mathbf{r}(u_p, v_p)$. Then the map $G : \gamma \rightarrow (u_p, v_p)$ is called a geometry field of surface S. From (u_p, v_p), the position and the normal of p can be obtained from the geometry image and the normal map. In general a geometry field is 4 dimensional. Only texture coordinates are stored in each entry. Therefore the size of a geometry field is manageable for current hardware. Figure 23 illustrates the concept of geometry field. A ray γ in $\mathbb{R}^3$ intersects a surface $\tilde{S}$ at a

point **p** on $\tilde{S}$ as shown in (a). $\tilde{C}$ is the bounding box of the surface, **s** and **t** are the intersection points of γ with $\tilde{C}$. **p** has a unique texture coordinates (u,v) as shown in (b). The position of **p** and the normal at **p** can be obtained from the geometry image (c) and (d), using the texture coordinates (u,v). A geometry field of a surface $\tilde{S}$ is map, the input is a ray γ in $\mathbb{R}^3$, the output is the texture coordinates (u,v) of the intersection point **p** between γ and $\tilde{S}$.

A surface and its geometry field are equivalent in the sense that the surface can be reconstructed from its geometry field, and the geometry field can be computed from the surface. By using a geometry field, real time ray tracing becomes a reality, because conventional expensive intersection testings are replaced by trivial table lookup operations. Our method makes contribution to conventional ray tracing with the combination of the advantage of rich precomputed information of light field and the regularity of geometry image.

Geometry field representation has several advantages. First, it effectively reduces the memory cost. The storage requirement will be extremely high, if the position, normal and color material information are stored for each entry. Instead, only the texture coordinates are stored, such that the size of a geometry field is within the memory capacity of current hardware. Second, due to the regular structure of a geometry field, it can be represented as a generalized texture. The whole rendering algorithm can be implemented in common graphics hardware. The time cost is independent of the geometric complexity of the scene, but dependent of the size of the geometry field.

Most real-time reflection methods make assumptions of the geometries of the relfectors or their relations with the scence, in contrast, geometry field handles arbitrary surfaces and the scences without any assumption. Furthermore, it offers ray surface intersections much more accurate than other algorithms. In experiments, we constructed geometry fields for complicated surfaces, such as Michelangelo's David head model with $100K$ faces. The inter reflections are computed using the GPU at frame rates as high as 60 FPS. The method can also be generalized for self-reflection and refraction also. To the best of our knowledge, our method is the first non-approximate method to achieve real time inter-reflection for surfaces with complex geometries and motions on general graphics hardware.

In the following section 2, we first provide detailed algorithms for constructing

geometry fields. Section 3 then shows how our system is implemented based on the recent advances of graphics hardware. After presenting and discussing our results in Section 4, we conclude and describe future research in Section 5.

5.2 Geometry Field

Although traditional ray tracing has the capability to achieve convincingly good quality, it suffers from lack of speed. A great deal of current researches have been targeted at improving its speed.

The common methods usually fall into several categories:

1. Improve the efficiency of all intersection tests.
2. Exploit temporal and spacial coherence.
3. Make use of high parallelism.

Our method adapts the same philosophy. The geometry field pushes all intersection tests into precomputation time and maximally reduces cost of intersection testing by table lookup at run time. The regularity of geometry images leads to large spatial coherence. The implementation on the latest programmable graphics hardware achieves the high parallelism.

This section explains the algorithms for constructing geometry fields from general meshes in detail. GPU implementation will be elaborated in the next section.

5.2.1 Construct Geometry Image

In our framework, all surfaces are represented as geometry images. First, triangular meshes are parameterized to a planar rectangle, then uniformly sampled on the planar domain to form a 2D array.

During the parameterization, distortions are unavoidably introduced. Geometric details will be lost if the parameterization schedule is not appropriate. In order to control the parameterization quality and determine the sample rate, we adapt the method of *conformal parameterization*.

Conformal parameterization preserves angles, and only area distortions are introduced. We determine the sample rate using the following algorithm.

1. Normalize the input mesh to be inside the unit cube.
2. Conformally parameterize the triangle mesh to a planar rectangle, using algorithms described in [23] and [24].
3. Super sample the original mesh on the planar domain to form a high resolution geometry image GF_0.
4. Subsample GF_0, and measure the geometric error using the algorithm in [23].

5. Repeat step 4, until the error exceeds the prescribed threshold.
6. Generate the normal map from the resultant.

Figure 23 demonstrates the geometry image of the David head surface, it is clear that all the geometric details and subtleties, such as the curly hair, are well preserved. Comparing the normal map on the texture domain, it is obvious that the geometry image is angle distortion free.

5.2.2 Construct Geometry Field

In general, a ray γ in $\mathbb{R}^3$ can be specified by the starting point and the ray direction, which is a unit vector, therefore the total ray space is 5 dimensional. Considering the limit of GPU texture memory, we represent a ray with (in, out) two points. Hence, the geometry field is a 4 dimensional subspace.

1. Outer-geometry-field
   To perform inter reflecting, we build a outer-geometry-field, which stands for the reflected ray directions using (in, out) on a unit bounding cube of the object. By default, we call outer-geometry-field as geometry field due to its broad usage. First, we construct a bounding box of the geometry image, then compute the intersection points of those rays that intersect the bounding box. Each such ray has two intersection points with the box, the entrance s and the exit t. We parameterize those rays using the point pair (s,t). We compute the intersection point of the ray with the geometry image closest to s and record the texture coordinates of the intersection point.
2. Inner-geometry-field
   To perform intra reflecting and refraction, we need a new geometry field,

which records the intersection point on the surface of outgoing ray directions either by reflection or refraction, using two intersects (in, out) on the object surface. First, we compute the intersection points of those rays that intersect the object surface. Each such ray has two intersection points with the surface, the entrance s and the first-meet intersect t. We parameterize those rays using the point pair (s,t). Since we compute the intersection point of the ray with the inner-geometry-field closest to s and record the texture coordinates of the intersection point, rays can be traced with multiple intra-reflection by repeated lookup in the inner-geometry-field.

In order to speed up the intersection calculation, we use the octree structure to subdivide the cube. All the computations are carried out in the local coordinates of the geometry image. The result is a $4D$ lookup table and the internal representation on the GPU is described in the next implementation section.

5.3 Rendering

The geometry fields and geometry images are generated offline in software. The run time rendering algorithms are implemented on NVidia's GeForce Quadro FX 4500 graphics card.

5.3.1 Data Structures

Each geometric object O_i is represented as a geometry field GF_i and a geometry image GI_i, which incorporates the position, normal, material and texture information.

Furthermore, each geometric object is normalized in the unit cube of bounding box. Points on the 6 surfaces of unit cube are sampled and enumerated in $1D$. So, the 4D geometry field is stored as a $2D$ texture, with (i, j) representing the indices for the in and out points on the unit cube.

In our implementation, all the data structures are loaded in the texture memory. Specifically, each normal map, position map in the geometry image are stored as two textures, and each geometry field as a single $2D$ texture.

One geometric object may have multiple instantiations, where different instances share the same data structures. Each instance has its own local frame and transformation matrices which transforms between the local frame and the world frame.

5.3.2 Algorithm Pipeline

Non-reflective or refractive surfaces are simply rendered using conventional OpenGL fixed pipeline functionalities. Reflective or refractive surfaces are rasterized with our geometry field based vertex shader and fragment shader. The vertex shader is conventional. Our fragment shader is capable of computing inter reflection accurately.

inter-reflection algorithm is as follows:

1. Calculate the eye ray in world coordinates.
2. Calculate the reflection ray γ in world coordinates.
3. Calculate the local shading $\mathbf{c}$ of the reflector surface.
4. Trace ray γ in outer-geometry-field to get reflected color $\mathbf{c}_r$, blend it with the local shading $\mathbf{c}$.

intra-reflection algorithm is as follows, which only differs from inter-reflection with Trace ray:

1. Calculate the eye ray in world coordinates.
2. Calculate the reflection ray γ in world coordinates.
3. Calculate the local shading $\mathbf{c}$ of the reflector surface.
4. Trace ray γ in inner-geometry-field to get inner reflected color $\mathbf{c}_r i$.
5. Trace ray γ in outer-geometry-field to get reflected color $\mathbf{c}_r$
6. Blend $\mathbf{c}_r$, $\mathbf{c}_r i$ with the local shading $\mathbf{c}$.

refraction algorithm is as follows,

1. Calculate the eye ray in world coordinates.
2. Calculate the refracted ray γ in world coordinates.
3. Calculate the local shading $\mathbf{c}$ of the reflector surface.
4. Trace ray γ in inner-geometry-field to get the first refraction position $\mathbf{c}_p i$.

5. Trace ray γ in inner-geometry-field to get the second refractive ray direction $\mathbf{c}_p i2$ and get the color for refracted image $\mathbf{c}_r$.
6. Blend $\mathbf{c}_r$ with the local shading $\mathbf{c}$.

The algorithm for tracing a ray is as follows,

Color TraceRay(Ray γ, ObjectId i, int depth)

1. For each object $O_j, i \neq j$ in the scene,
    1. Transform the incoming ray starting point and the ray direction from the world coordinates to the local frame of O_j.
    2. Check if the ray hits the bounding box of O_j (the unit cube).
    3. If the ray hits the box at points s and t, convert s and t to the indices of the geometry field GF_j.
    4. Lookup the geometry field GF_j to obtain the texture coordinates (u_j, v_j) of the intersection point.
    5. If (u_j, v_j) is valid, lookup the position from the geometry image GF_j, compute the depth d_j. Otherwise, set $d_j = \infty$.
6. Choose the nearest $(u_k, v_k), k = min_j\{d_j\}$. If (u_k, v_k) is valid, look for the position, color, normal of the intersection point from GI_k, compute the shading and blend it to the local color $\mathbf{c}$, compute the next level reflected ray γ_r.
7. If the recursion depth exceeds the limit, return $\mathbf{c}$ else call TraceRay(γ_r, k, depth+1) to get the reflected color $\mathbf{c}'_r$.
8. Blend $\mathbf{c}$ and $\mathbf{c}_r$. Return the blended color.

In current graphics hardware, recursive functions are expended by the Cg compiler, the recursion depth is limited by the upper bound of fragment program length. In our implementation, the recursion depth is 2.

5.3.3 Packing Geometry Field

Each ray γ intersecting the unit cube can be represented by the two intersection points, denoted as s and t, where s is the entrance, t is the exit.

The points on the unit cube are sampled and arranged as a two-dimensional array. Figure 24 illustrates our method for indexing the samples on the cube. Suppose the sample rate is n along each dimension, the 6-face is ordered and rearranged

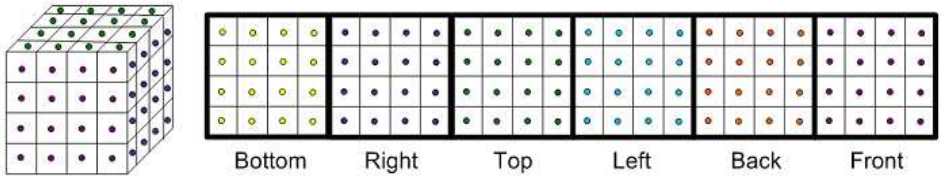


Figure 24: The points on the unit cube are sampled and arranged as a two-dimensional array.

to form a big rectangle of size $6n \times n$. Suppose the entrance point s is on face $k \in \{0,1,\cdots,5\}$, the coordinates of s on the face is (x,y), then the indices (x_0,x_1) of s are calculated as $(x+kn,y)$.

Similarly, we calculate the indices for the exit point e, denoted as (x_2,x_3). Then we use (x_0,x_1,x_2,x_3) as the indices of the ray to lookup the geometry field.

In our implementation, the geometry field is represented as a $2D$ texture. Ray coordinates (x_0,x_1,x_2,x_3) are converted to the texture indices $(6nx_1+x_0,6nx_3+x_2)$.

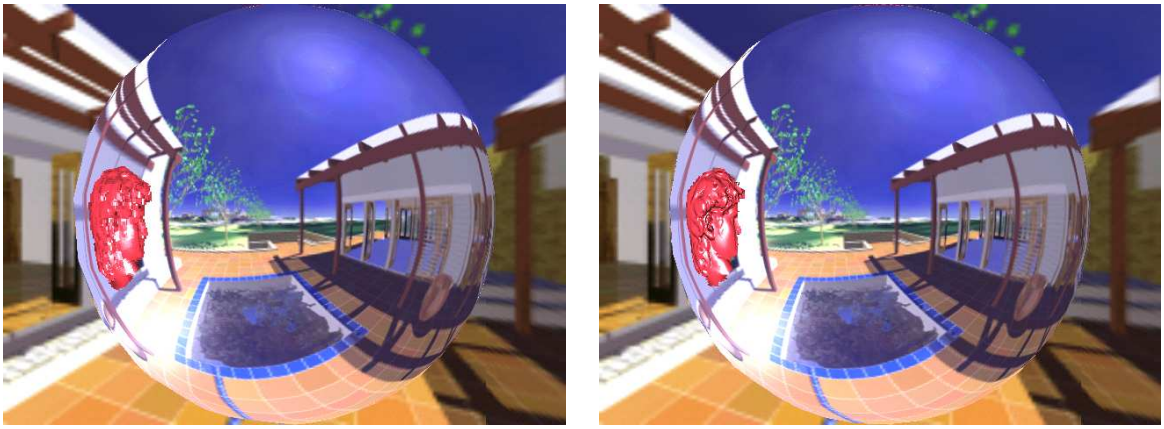
5.3.4 Lookup in Geometry Field

In current hardware, the largest $2D$ texture size is $4K \times 4K$, and the total sample points on the bounding box should be no more than $4K$. Assume the sample rate along each dimension is n, then the total samples $6n^2$ should be no more than $4K$. The upper bound of the size along each dimension is 24.

The rays are sparsely sampled, motivating us to develop a 4 dimensional interpolation scheme to remove artifacts. With speed in mind, we adapt simple convolution kernels (currently linear) for interpolation. Given a ray in the ray space with coordinates (x_0,x_1,x_2,x_3), we define $x_i^0 = floor(x_i)$ and $x_i^1 = ceil(x_i)$ and the weights $w_i^0 = x_i^1 - x_i$, $w_i^1 = x_i - x_i^0$, then the interpolation of a function f defined on the ray space is defined as

$$f(x_0,x_1,x_2,x_3) = \sum_{i,j,k,l=0}^{1} f(x_0^i,x_1^j,x_2^k,x_3^l) w_0^i w_1^j w_2^k w_3^l.$$

In our implementation, function f is the (u,v) texture coordinates. Figure 25



(a) Nearest neighbor sampling (b) 4D linear interpolation

Figure 25: Comparision between the rendering results using nearest neighbor and $4D$ linear interpolation to perform lookup in the geometry field.

demonstrates the rendering quality difference between different lookup methods using nearest neighbor and linear interpolation schemes.

5.4 Experimental Results

This section demonstrates the experimental results for our real time inter reflection algorithm based on geometry fields.

5.4.1 Preprocessing

In our experiments, we tested our algorithms on several triangular meshes. The geometric complexity is from several thousand faces to tens of thousands of faces. The complexity information and the performance of our algorithms are detailed in Table 5. The experiments are applied on the Windows platform with a $3.6GHz$ CPU and $3G$ RAM. All geometry fields are of size of $(24 \times 24 \times 6)$.

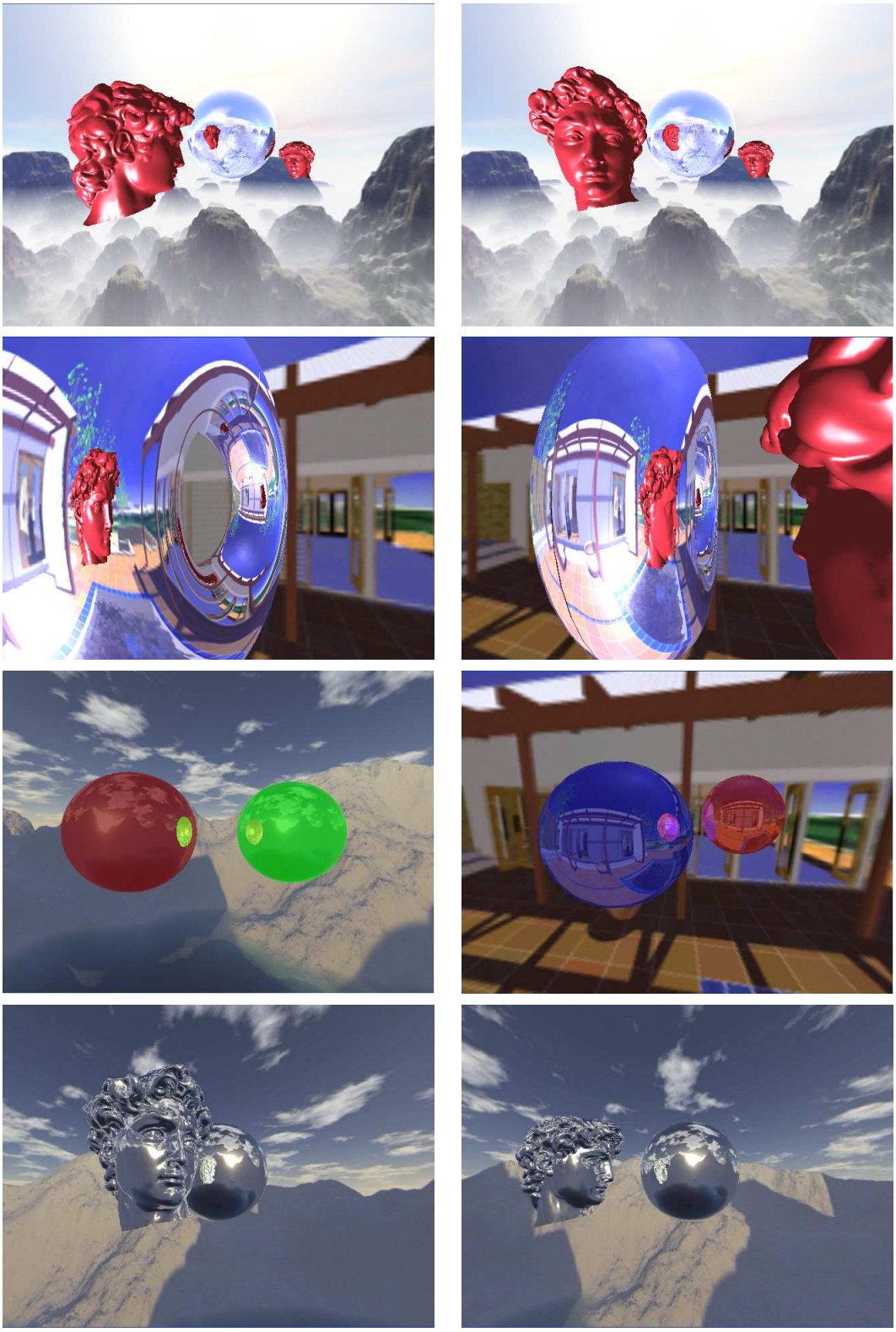


Figure 26: Real time rendering results. User can interactively manipulate the camer position, the relative spactial relations among the geometric objects in the scene. The inter reflections are rendered in real time.

5.4.2 Real Time Rendering

Figure 26 demonstrates our real time rendering inter reflection results.

In the first row, there is a reflective sphere and two instances of the David head surface. The background is rendered using a environment map.

The view point, the relative positions of the David head surfaces and the mirror ball are changed interactively by user inputs. The mirror images of the David head on the mirror ball retain all geometric details. The distortions of the mirror images are calculated accurately without any approximation. The frame rate varies from 50 to 60 frames per second.

In the second row, the scenary is replaced by a more complicated torus surface. The camera is moved close to the torus, so the mirror images of the David head surfaces can be examined in detail. The complex geometric features of the hair in the mirror are rendered with high fidelity. Although the reflector surface itself contains more complicated geometry, the frame rate is still kept at the same level. This demonstrates the fact that the cost for table lookup of the geometry fields is determined by the size of the geometry fields, and is independent of the geometric complexity of the surface.

Inter-surface reflections between two mirror balls are illustrated in the third rows of the Figure 26. From the snapshots, the mirror image of the red sphere is shown on the blue sphere. In the center area of the mirror image, the mirror image of the blue ball on the red ball is also recognizable. This demonstrates the second level inter reflection. Although the ray tracing depth is increased, the frame rate is still around 60 FPS.

Table 5: Performance (in sec) of our preprocessing algorithm.

Surface	Triangular Faces	Parame-terization	Geom. Image	Geom. Field
Sphere	1,922	15	0.8	420
Torus	7,938	34	2.2	420
Female Face	50,000	130	9.5	432
Male Face	80,072	200	16	504
David Head	101,144	250	20	672

The last row demonstrates the inter reflection between the sphere and the David head surface. The geometric complexity of the David head surface doesn't affect the performance.

Figure 27 shows our real time rendering results for refraction and intra reflection results.

The first row shows a single refraction result with a sphere on the background of the Nvidia lobby and a transparent kitten floating in a yard with green trees in the background. In the second row, full refraction rendering results are illustrated. With full refraction on both surfaces of the sphere, visual image result shows background upside down. The frame rate for refraction is about 50 FPS.

In the third row, self-reflection rendering results are shown. In order to closely watch the subtle difference made by self-reflection, we show only the intra-reflection result of a kitten and a sphere in the left image. Obviously, the kitten has self reflection between its head and back, while the convex sphere has no intra-reflection. In the right image, it demonstrates the rendering results combining with the intra and inter object reflection. The frame rates keeps above 50 FPS.

The experimental results convince us that the real time rendering method based on the geometry field is capable of simulating inter-surface reflection, refraction and intra-surface reflection.

5.5 Conclusions

In this chapter a novel framework for real-time inter reflection by ray tracing based on the geometry field was proposed. A geometry field is a combination of a light field with a geometry image, and represents the intersection point of a surface with an arbitrary ray as a $4D$ lookup table.

Conventional intersection testing in ray tracing is replaced by a look-up in a geometry field, such that real time inter-surface reflection is achieved on current graphics hardware. The method is efficient, accurate and simple. Experiments have shown the high quality of reflection results and fast rendering speed in several examples.

Applications of geometry fields include refraction, reflection between objects

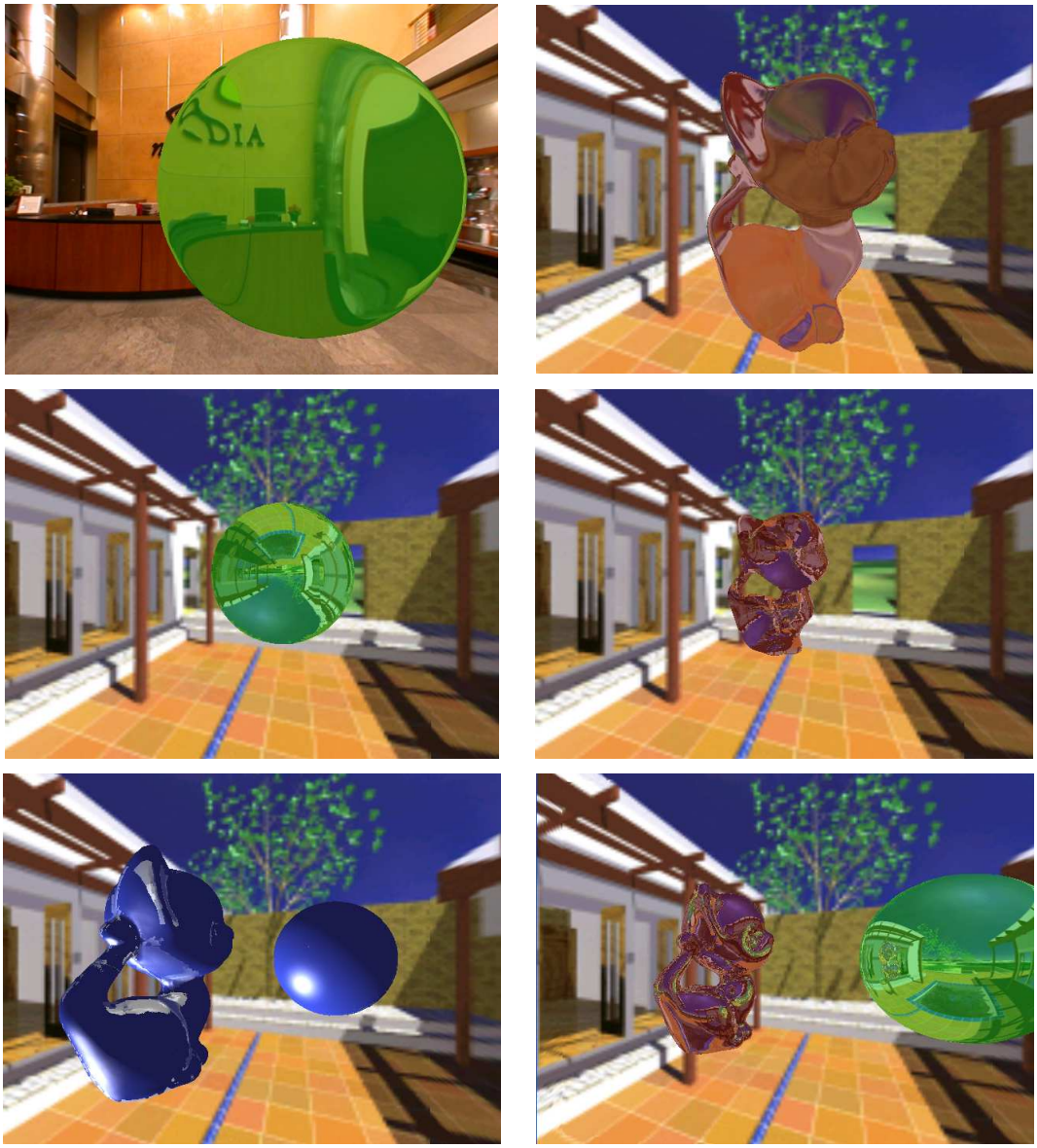


Figure 27: Real time refraction and self-reflection rendering results. Upper two rows show the refracted image. Lower row represents the self-reflecting images.

and self-reflection. We have implemented a full-featured ray-tracing based on geometry fields in real time. Geometry fields also have great potential to improve the efficiency for other sophisticated global illumination methods to real time.

Chapter 6

Breast Ultrasound CT Refraction Correction

A significant obstacle in the advancement of Ultrasound Computed Tomography has been the lack of efficient and precise methods for the tracing of the bent rays that result from the interaction of sound with refractive media. In this chapter, we propose the use of the Fast Marching Method (FMM) to solve the Eikonal equation which governs the propagation of sound waves. The FMM enables us to determine with great accuracy and ease the distorted paths that the sound rays take from an emitter to the receivers. We show that knowledge of the accurate path proves crucial for an object reconstruction at high fidelity and accurate geometry. We employ a two-phase approach with an iterative method, SART, to faithfully reconstruct two tissue properties relevant in clinical diagnosis, such as mammography: speed of sound and sound attenuation. We demonstrate our results by ways of a newly designed analytical ultrasound breast phantom.

6.1 Introduction

Ultrasound computed tomography (UCT) has a long history and particular promise in the imaging of the breast. However, the reconstruction of these images poses significant challenges. UCT is susceptible to refraction effects, making it difficult to reconstruct images faithfully. The acoustic ray direction is bent when

ultrasound passes from one medium to another medium with a change in the acoustic index of refraction, according to Snell's law for refraction. For example, the large subcutaneous fat layer in the breast [67] [54] causes a refractive effect that can significantly distort the ultrasound ray direction and eventually cause spatial distortion and intensity artifacts in the resulting images. Diffraction is another complication typically addressed by diffraction tomography methods [33] but it is based on the weak scattering assumption [50], which is violated by the strongly refracting fat layers in the breast. In this work, we focus on correcting the artifacts stemming from refraction. Previous work has either not modeled bent rays at all or has inadequately eliminated bent ray distortion effects and failed to faithfully reproduce tissue properties in UCT. Furthermore, prior methods have been computationally expensive, limiting their extendibility to three dimensions.

To advance the state of the art in these respects, we introduce the concept of wave-based ray propagation models into UCT imaging, accurately taking into account the refractive phenomena. For this, we model the eikonal equation, which governs the movement of a wave front from emitters to receivers, using the Fast Marching Method (FMM), described by Sethian [61]. With this method, the wave arrival time for each grid point can be extracted, and the accurate ray direction for an arbitrary point can be derived by searching for the minimum path in the Time-Of- Flight field between the point and the emitter. The FMM has become quite popular in recent years in computer graphics and computer vision, enabling accurate distance transforms, segmentation shape recognition, and others. In this chapter, we demonstrate that the FMM also represents a promising method for the efficient and accurate modeling of the propagation of acoustic waves in a refractive media.

This chapter is structured as follows. In Section 2, we provide the theoretical background on the models we propose and Section 3 describes the implementation details of our algorithm for the modeling of nonlinear rays for the reconstruction. Section 4 then presents and discusses our experimental results. Finally, we conclude and describe future directions of research in Section 5.

6.2 Theoretical Background

6.2.1 Reconstruction Algorithm

Classical tomography reconstruction algorithms using Filtered Backprojection are based on the Radon theorem and can not take into account bent rays. Therefore, similar to other UCT researchers, we employ an algebraic reconstruction approach, SART [2]. Given the projection data p_i, SART updates a pixel v_j in iteration k according to the following equation:

$$v_j^k = v_j^{k-1} + \lambda \frac{\sum_{p_i \in P_\Phi} c_i w_{ij}}{\sum_{p_i \in P_\Phi} w_{ij}}; c_i = \frac{p_i - \sum_{i=1}^{N} w_{il} v_l^{k-1}}{\sum_{l=1}^{N} w_{il}} \quad (1)$$

Here the w-terms relate the pixels to the data and are determined by the interpolation function. The correction/update factor ci is computed by subtracting the result of a discrete ray integration (within the grid constructed at iteration (k-1)) from the physical integration acquired at receiver i. In our case, due to the refraction effects, the rays are non-linear. SART is a block-based algorithm, i.e., a grid update occurs after all rays for a given source (emitter) have been traced and the correction factors computed.

6.2.2 Solving the Eikonal Equation with the FMM

As discussed above, our approach advocates an alternative way to solve the bent-ray problem by directly simulating the acoustic sound wave propagation. Bent rays can be computed by solving the eikonal equation [61]:

$$(\frac{\delta t}{\delta x})^2 + (\frac{\delta t}{\delta y})^2 + (\frac{\delta t}{\delta z})^2 = \frac{1}{F^2(x,y,z)} \quad (2)$$

on a discretized grid of points. Traditionally, equations such as equation 2 are solved by iterative methods, which can be computationally expensive. To solve the eikonal equation more efficiently, we employ the Fast Marching Method (FMM), originally proposed by Tsitsiklis [68]. The FMM is related to Dijkstra's method [16], which is a classical algorithm for identifying the shortest path in a network of links. The FMM is a single-pass, upwind finite difference scheme, which produces

the correct viscosity solution to the eikonal equation. It depends on a causality condition based on the ordering of the upwinding [61].

In equation 2, F is called the speed term and is a measure of the local sound conductance properties. The FMM computes for every voxel (x,y,z) the time $T(x,y,z)$ at which the wave has traversed it. As the wave front proceeds across the grid, the FMM selects the voxel (x,y,z) in the narrow band of voxels (situated immediately upwind from the current wave boundary) which minimizes the time increment, given the values of its neighbors and their speed values. The result of the FMM is the Time Of Flight (TOF) image. There is one such image for each emitter.

The original FMM solves the eikonal equation by using only first-order finite difference. This will lead to inaccuracies at high curvature boundaries. For a more accurate approximation of equation 2, we use the High Accuracy Fast Marching Method (HAFMM) [4] . It employs a second-order approximation to the partial derivative in equation 2, such as

$$\frac{\delta t}{\delta x} = \frac{3t(x,y,z) - 4t(x-1,y,z) + t(x-2,y,z)}{2} \tag{3}$$

but it also requires accurate second-order estimates for initialization around the propagation seed points (emitter locations).

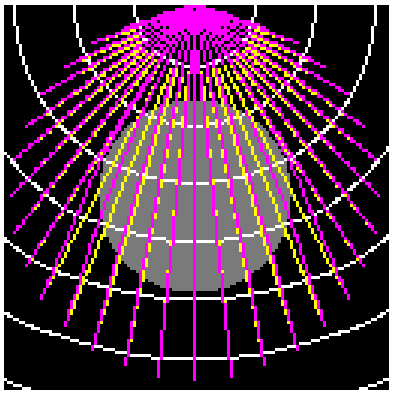


Figure 28: Curved-ray (red) and straight ray(yellow)with the FMM

Once the propagation is complete, we use the resulting TOF image to calculate the path of the rays from the receivers back to the emitter. The TOF image allows

us to locally compute the ray direction vectors, given by the TOF image gradients, ensuring that a given ray will not miss the emitter. Our method thus eliminates the need for the ray linking and path assembly of earlier approaches. Figure 28 shows the acoustic ray paths from 31 receivers, which are distorted when passing through the object. White curves represent the wave front at different times. The yellow lines are the straight rays between emitter and receiver and the red lines represent the curved-rays obtained via HAFMM.

6.2.3 Ultrasound Breast Modeling

In the area of computational tomography, there are a number of existing breast phantoms [8] [64]. However, they are usually too complex, and some of the structures are not perceptibly differentiated in UCT image. Therefore, a simple, numeric UCT breast phantom is proposed, as shown in Figure 29.

(a) anatomy(from info.med.yale.edu)

(b) phantom(left:vertical slice; right:horizontal slice.)

Figure 29: Breast anatomy and phantoms.

Our UCT breast phantom is modeled using a half ellipse, and is composed of two main layers: an outer thick layer of fat and an inner layer of tissue. In the inner layer of the tissue, several lesions are shown as small ellipses, and smaller

abnormalities modeled as tiny spheres are inserted. Keeping the shape unchanged and adjusting the values of the phantoms, we can simulate a sound velocity (sound conductance) phantom and an attenuation phantom separately. See Table 6 for the clinical values used in the phantoms. Both the sound velocity and the attenuation properties of tissue provide valuable diagnostic information. The analytical geometric description of the elliptical primitives allows for easy analytical modeling of refracted ray paths and their path integration in the simulation of projection data. We have not used real data at this time, in order to isolate the aliasing and distortion effects purely due to the non-linear, refracted rays (as opposed to higher-order scattering noise).

6.3 Methodology

Our novel contribution is to combine *SART* with FMM to find the accurate ray directions by wave propagation. In this way, we can avoid the complicated bent-ray computations that previous UCT reconstruction algorithms had to deal with, replacing them with the simple and linear computations embodied by the FMM.

In this research, we implemented both FMM and HAFMM. We use a binary heap to quickly find the voxel with the smallest postulated wave arrival time in the narrow band of the advancing wave front. The wave arrival value of this voxel is then written to the Time-Of-Flight (TOF) image, its neighbors and their arrival times are updated in the heap. Note that a translation table with double pointers, as is further described in [61] [4] is used in order to quickly map the spatial domain voxels to the heap voxels.

Our framework itself can be decomposed into a two-phase algorithm. In phase 1, we iteratively reconstruct the sound velocity (SV) image from the TOF data collected at the receivers, and in phase 2, we use this SV image to guide the non-linear

Table 6: Our phantom's breast ultrasound properties

Ultrasound Properties	Tissue	Fat	Large Lesions	Small Lesions
Velocity	$1475m/s$	$1375m/s$	$1560m/s$	$1530m/s$
Attenuation	50	15	$60or30$	70

rays for the iterative reconstruction of the sound attenuation (SA) image from the attenuation data collected at the receivers. Both phases use SART as the iterative reconstruction engine. Note that the SA image is easy to reconstruct once an accurate SV image is available to guide the distorted rays, provided the gradients are faithfully reconstructed using good interpolation filters. Our algorithm proceeds as follows:

For the SV update step, we use the relationship $v = d/t$. Here, d is the diameter of a spherical pixels (we assume spherical pixel to achieve direction independence and use $d = 1$). The following equation is employed:

$$SV^{k+1} = \frac{d}{\frac{d}{SV^k} + \sum_{rays} \Delta TOF} \tag{4}$$

One SART iteration completes after all emitters have been processed once, and the iterations continue until the difference between the TOF image at the receiver positions and the collected TOF data there becomes smaller than a threshold. Usually, this requires 3 to 4 iterations. The reconstruction of SA image is similar, only now the SV image remains constant and with it the ray paths and their lengths. Note, the algorithm requires the estimation of good gradients. For this, we employ B-splines, which have previously shown to work well in refractive media [14] [40].

The velocity of a grid point, stored in the SV image and used in the wave propagation step of the reconstruction, depends heavily on the reconstructed value at that point obtained from the previous iterations. To obtain the accurate speed value, we investigated two different approaches: fixed speed update and data-driven speed update.

Fixed speed update means that the speed update is applied directly to the pixel, without further scaling.

$$F = kSV \tag{5}$$

Data-driven speed update means that the speed update is normalized in the following equation:

$$F = k(F_{min} + \frac{SV - SV_{cur-min}}{SV_{cur-max} - SV_{cur-min}}(F_{max} - F_{min})) \tag{6}$$

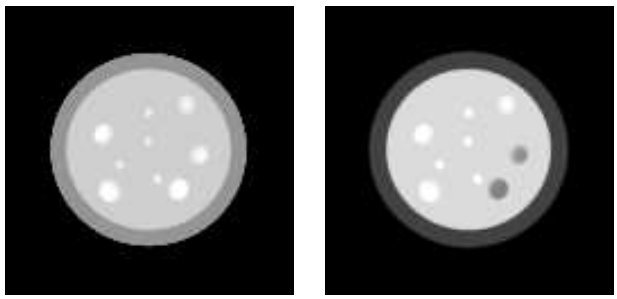
In equations 5 and 6, k is a constant scale factor, $SV_{cur-min}$ and $SV_{cur-max}$ are the speed extrema of the current iteration, F_{min} and F_{max} are the inherent speed extrema of the object, and F is the resulting propagation speed.

The fixed speed update strategy is intuitive, considering the fact that the values in the SV image represent material properties, and acoustic rays always have the same speed in a specific material. However, it can suffer from the problem that the shape of the organ is distorted by the first few iterations speed value, when the correct value has not been constructed yet.

Data-driven speed update is introduced to solve this problem in the iterative reconstruction algorithm. In this method, we assume that in the initial iterations the pixel's absolute value may not be accurate but the overall geometric information has been quickly formed and recorded in the grid point's relative values. The normalization ensures a fast ascent of the solution at early iterations when SV values are small.

For the construction of the SA image, the TOF image can either be computed beforehand, or on the fly when storage is excessive, using the FMM on the reconstructed SV image. The input data are now the collected attenuation data, one set for each emitter, and SART proceeds as usual for each randomly chosen emitter position, using the rays guided by the corresponding TOF image, but now updating the attenuation volume.

6.4 Experiments and Results



(a) velocity (b) attenuation

Figure 30: Ultrasound phantoms.

Our experiments are based on a simulated computer phantom with a matrix

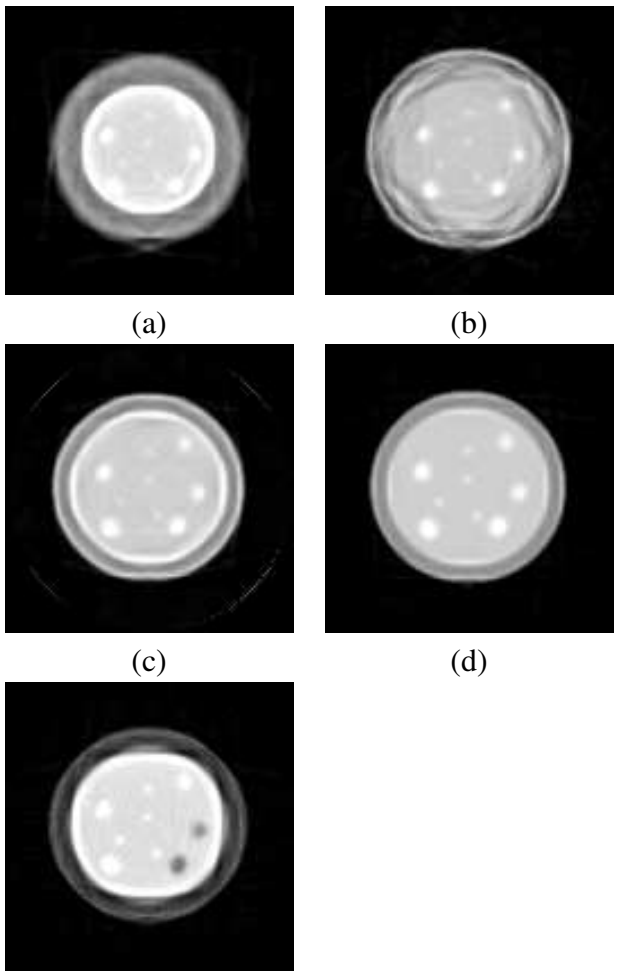


Figure 31: Reconstructed images: (a) straight ray SV image, HAFMM; (b) nonlinear ray SV with fixed speed, HAFMM; (c) nonlinear ray SV with relaxed speed, FMM; (d) nonlinear ray SV with relaxed speed, HAFMM; (e) attenuation image.

size of $128x128$, and the quantitative parameters are given in Table 6. The phantoms are shown in Figure 30, holding lesions with diameters from 2 to 8 pixels. The ultrasound TOF and attenuation phantoms share the same geometry. Our simulation assumes a spherically radiating sound emission, using 256 emitter/receiver positions in a circle. The resolution of the reconstructed image matches the size of the phantom images. More emitter/receiver pairs enable higher resolution. Reconstruction costs about 60 seconds for 3 iterations using a $2.8GHz$ Pentium 4.

For the velocity phantom, the reconstruction results are given in Figure 31(a)-(d). With HAFMM, the reconstructed image obtained with linear rays is shown in (a) and non-linear rays in (b) and (d). For the non-linear ray reconstruction, we further compare the results obtained for the different speed strategies: fixed speed in (b) and data-driven speed in (d). We calculate the error as the average absolute intensity difference for all grid points between the reconstructed image and the original phantoms. The use of only straight rays distorts the image with an error of 25%. Because refraction is ignored, the size of the phantom's regions grows or shrinks dramatically. When correcting for refraction, the speed strategy has a large impact on the resulting image accuracy. The data-driven relaxed speed results in a better estimation of the original image (error of 3%), while the fixed speed strategy overcorrects for refraction with an error of 19%.

The image achieved using FMM is shown in Figure 31(c), with the error of 10%. HAFMM in (d) is more accurate than FMM in (c), since the HAFMM includes second-order curvature information when solving the eikonal equation. This means that boundaries where refraction occurs are better determined.

Finally, the attenuation phantom reconstruction results are presented in Figure 31(e). It is based on the nonlinear ray paths determined in the SV phantom reconstruction shown in Figure 31(d), using the HAFMM. The attenuation image recovers both intensity and shape accurately with an error of less than 7%.

6.5 Conclusions and Future Work

In this paper, a new method for UCT image reconstruction taking into account refraction was proposed. The key observation behind this method is that the upwinding Fast Marching Method (FMM) can determine the first-arriving phase in a

continuous medium. It also provides computational efficiency in determining the accurate paths of the non-linear rays. We proposed two speed mechanisms for the FMM to trace the refractive rays: fixed speed and datadriven relaxed speed. Our method is applicable in any UCT imaging situation in a moderate refractive media. UCT simulations, using a synthetic breast phantom, have demonstrated that the continuous wave propagation in HAFMM with data-driven relaxed speed achieves an accurate determination of the non-linear ray paths and provides much better fidelity in the image reconstruction. The reconstructed image quality improves by an order of magnitude compared to the pure straight ray method.

Chapter 7

GPU Acceleration for UCT

7.1 Introduction

Compared to traditional X-ray breast mammography, ultrasound has the important advantage that no radiation dose is delivered to the patient. In addition, tomographic methods, as opposed to single projections of a compressed breast, have the advantage that lesions can be better localized in space. A modality combining all of these advantages is transmission ultrasound CT (UCT). Here, the patient submerges the breast into a water bath (which is far more convenient than the compression needed for X-ray breast mammography) and a ring transducer array (with $100s$ of elements) axially positioned around the breast acquires the time-of-flight and attenuation data used for CT reconstruction, one set of receiver data for each emitter position. For $3D$ reconstruction, one may either use a multi-ring device, or one may axially translate or spiral the ring and repeat acquisition. Such systems have been described [17].

The fact that acoustic rays suffer from severe refraction effects in the imaged tissue has hampered the direct use of traditional analytical back-projection or iterative projection/back-projection frameworks to achieve a practical UCT solution. Existing approaches so far have been unable to model the resulting curved rays efficiently and accurately, and this has limited the image resolution that can be achieved. While recent work in iterative UCT [18] shows promise and discusses reflection and refraction effects, the reconstruction algorithms discussed there are still

limited to the straight-ray assumption. In earlier work, several researchers (e.g. [3] have explored the use of general ray-tracing, via ray-linking and ray rebinning, in P/BP UCT to improve the image quality. Denis [14] compared several methods for ray-tracing, showing that substantial improvement over straight ray methods can be achieved for moderately refracting fields. Yet, the more recent work typically avoids the expensive direct ray simulation. For example, Quan [55] reports that a simulation of 256 detectors would take weeks to complete on a desktop PC, requiring supercomputers for UCT reconstruction. They therefore use a sparse matrix solver to accomplish the reconstruction in reasonable time. Finally, an alternative method for UCT is diffraction tomography [50]. It is based on the weak scattering assumption, which, however, is violated in case of the breast, due to its strongly refracting fat layers.

Essentially, sparse matrix solver-based and Fourier-space methods do not model the ray physics in a direct manner, but through some intermediate representation, which cannot easily capture the local interactions of the ray with the medium. On the other hand, a complete simulation of the wave equation is prohibitively expensive. Yet, we [41] have shown in Chapter 6 fast wave-front propagation methods, such as the Fast Marching Method (FMM) [61], can model the governing Eikonal equation well, taking into account refraction effects and allowing the resulting curved rays to be accurately modeled for both speed and attenuation in their native physical domain. We aim to resolve lesions of single pixel-size (data resolution), various shapes (speculated, lobulated, sharp, circumscribed), and a pixel-level distance among lesion clusters. At an US frequency of $7.5MHz$ one can reach depths of $16cm$, exceeding the requirements for breast imaging, allowing a sub-mm resolution of less than $0.3mm$.

While the FMM-based approach is far more efficient than a full wave equation simulation, we find that it is still not efficient enough to warrant deployment in a clinical setting. Thus, we look into the use of commodity high-performance hardware for parallel program execution, such as multi-core CPUs, programmable commodity graphics hardware (GPU), and processor clusters. However, the inherently sequential mechanism of the FMM method poses certain trade-offs with regards to the architecture used (and also cost), which we will show can be balanced by use of other equivalent fast wave front tracking algorithms, such as the Fast Sweeping

Method (FSM) [9] and the Fast Iterative Method (FIM) [32]. These considerations and their results are the topic of this Chapter, with the overall conclusion being that UCT based on fast wave-front tracking has excellent potential for real life clinical deployment

7.2 Methods

The algorithm put forward in Chapter 6 uses SART (Simultaneous Algebraic Reconstruction Technique) [2] as the basis of the UCT reconstruction algorithm:

$$v_j^k = v_j^{k-1} + \lambda \frac{\sum_{p_i \in P_\Phi} c_i w_{ij}}{\sum_{p_i \in P_\Phi} w_{ij}}; c_i = \frac{p_i - \sum_{i=1}^{N} w_{il} v_l^{k-1}}{\sum_{l=1}^{N} w_{il}} \quad (1)$$

The propagation of the US wave front in 3D is modeled by the Eikonal equation:

$$(\frac{\delta t}{\delta x})^2 + (\frac{\delta t}{\delta y})^2 + (\frac{\delta t}{\delta z})^2 = \frac{1}{F^2(x,y,z)} \quad (2)$$

Here, the speed term F is a measure of the local sound conductance properties. To solve the equation for F one must find the speed trajectory for each ray that minimizes the time cost function from the source. This is iteratively solved using the SART. The forward projection computes for every voxel (x,y,z) the time $T(x,y,z)$ at which the sound wave, originating from the emitter, has traversed it. We call this the time-of-flight (TOF) image, which includes the TOF estimates of the receivers them-selves (used to correct the grid corrections). This TOF image allows a local computation of the ray direction vectors from all receivers back to the single emitter using the TOF image gradients, estimated via a high-quality filter. This crucial step ensures that a given curved ray will not miss the emitter, eliminating the path assembly overhead of earlier approaches. More formally, this Eikonal-SART algorithm is described as follows:

1. Construct the TOF field as described above.
2. Compute the corrections c_i by subtracting the Eikonal-computed TOF field at the receivers from the true TOF data. Calculate the normalization weight by tracing curved rays back to the emitter using the TOF image and a field of unit voxels.

3. Update the speed terms of the voxels $v_i j$ by the reciprocal of the c_i, again using the TOF image to determine the curved ray paths.
4. Normalize the $v_i j$ by the weights (also accumulated in step 3).
5. Randomly select another (unused in this iteration) emitter and return to step 1.

In Chapter 6, we also described a data-driven relaxation factor to control the voxel updates, which yields superior results over the fixed relaxation factor λ typically used in SART. In fact, our UCT reconstructs two tissue properties, which gives non-redundant information. Once the sound velocity (SV) image is reconstructed it is utilized to guide the non-linear rays for reconstructing the sound attenuation image using attenuation data.

7.2.1 How to perform refractive UCT efficiently: general considerations

Each iteration of Eikonal-SART is performed for each of the M emitters: (i) one forward wave propagation across a lattice of N^3 voxels, and (ii) one backward projection delivering the updates to the N^3 voxels via $M-1$ non-linear rays which are guided by the TOF field computed in step (i). Let us assume we use $M = N^2$, that is, N transducers for each of the N rings (we shall neglect for now that the reduced axial dimension of the breast would allow far less rings to be used in practice). This makes for a slightly undetermined equation system which has been shown to be sufficient in iterative reconstruction scenarios. For the raytracing in the back-projection step, we have N^2 rays of length $O(N)$ each, but the traversal involves only local and independent computations at each step, at low computational overhead. Thus the overall complexity for one back-projection is $O(N^3)$. The wave-front tracking in the forward projection also updates N^3 voxels to yield the TOF image (and advances the front), but the selection of the advance-voxel has complexity $O(N^3)$ since one must consider all voxels within the wave-front's narrow band (which may be arbitrarily complex). Thus the complexity for wave front tracking is $O(N^6)$. Raytracing is quite amenable to parallel implementation and speedups on the order of $1-2$ magnitudes can be obtained on various parallel platforms, such as GPUs (e.g. [52]). The independence of the individual rays affords

a relatively fine-grained parallelism. In contrast, the wave front tracking does not exhibit much parallelism, since it is an inherently sequential algorithm where the selection of a voxel into the front depends on all previous selections. One method to reduce the wave front tracking overhead per emitter is to divide the $3D$ grid into N $2D$ slices. This strategy gives good parallelism on a coarse-grained level, due to the independence among slices. There is no communication overhead between slices, no need for data distribution as each slice can reside on a separate processor, and no synchronization delay because each computation proceeds independently. While the $2D$ slice-based reconstruction may lead to errors due to out-of-plane wave propagation, our experiments indicate (see Section 3) that the reconstructions using the $2D$ slice decomposition are quite similar to the corresponding fully-$3D$ reconstruction, but can be obtained at much higher speeds. But even with this decomposition there are vast differences in the level of speedup obtained with different Eikonal equation solvers (all using physical-space wave front tracking), also as a function of computing platform, as is discussed next.

7.2.2 Numerical Eikonal equation solvers and their parallelization potential

The Fast Marching Algorithm (FMM) algorithm [61] solves the Eikonal equations in one pass, tracking the evolution of an expanding front. A key observation of FMM is that the solution of Eq. (2) at each grid point depends only on the smaller values of neighboring points. Thus the solution of the equation can be built in the order of increasing arrival time of points. Here it is essential to quickly locate the voxel with the smallest value in the active "narrow-band" of n voxels. Typically a heap is used for this, with an update (insertion) time of $O(\log n)$. Sapiro [80] proposed a faster algorithm using an untidy priority queue, which replaces the heap's tree by a table, reducing the update complexity to $O(1)$. This, however, also introduces extra errors caused by misorderings inside the queue. We employed the (min-)heap data structure for accuracy, using double-linking between th lattice and heap for speed. The FMM is not naturally accelerated on fine-grained parallel systems. Neither the min-heap data structure nor the priority queue, which enables

fast speeds on the CPU, encourages parallelism, since they all sequentialize computations and make them globally dependent. The slice decomposition described in Section 2.1 favors coarse-grained parallelism, assigning projections of different slices to different threads or processors and running them concurrently. GPUs, programmed with CUDA, are well suited to compute this type of problem, but one must be aware of the large cache memory requirements of the fast min-heap operations. Section 3 shows that this poses a limiting factor for GPUs, where the register, cache and global memory are limited.

The Fast Sweeping Method (FSM) algorithm [82] applies Gauss-Seidel iterations with alternating sweep orderings. The key is to follow the solution characteristics and update points in that order, i.e., the sweeping direction should fit the real information propagation. In two dimensions, the sweep order is according to these four loops:

1. $i : 1..nx, j : 1..ny$
2. $i : nx..1, j : 1..ny$
3. $i : nx..1, j : ny..1$
4. $i : 1..nx, j : ny..1$

where i and j are the sweeping point positions in the x- and y-directions, respectively, and nx and ny are the number of points in the x-direction and y-direction, respectively. The FSM is amenable to parallelization, since it follows the causality along the solution characteristics in a parallel manner. It can compute multiple sweeping directions simultaneously. In addition, it can divide the whole domain into subdomains and execute sub-domain computations in parallel. It is well suited for multi-processor CPU systems or clusters. However, since in each sweep thread in a subdomain the computation is sequential, it is not easy to achieve fine-grained parallelism.

The Fast Iterative Method (FIM) algorithm [32] solves PDEs of H-J (Hamilton-Jacobi) equations on parallel architectures. The Eikonal equation is a special case of the H-J equation. It uses Godunov upwind discretization of the Hamiltonian [60] and updates the value on a narrow band iteratively. The FIM allows multiple updates per point, by re-inserting points into the narrow band, called the active list (AL). The FIM gives values of zero to the source and inf to all other

points. All neighbors of the source points form the AL. In the update loop, the FIM calculates the Godunov Hamiltonian for all AL points and updates their distance values if the newly calculated value is less than the current. If an AL point has converged, it is removed from the AL. All neighbors of the AL points will also update and if their value is less than the current, they will be added to the AL. This continues until the AL is empty. This algorithm maps well onto GPUs with parallel SIMD processors due to its fine-grained level parallelism, updating all AL points in parallel. In order to fully use the coherent memory access and control flow, our GPU implementation uses data-grouping as the primitive execution unit. Each point inside a data-group is executed as a thread, all threads for the same data-group combine into a block, and all blocks combine into the CUDA computational domain as a grid. The CPU-algorithm AL becomes the Active Block List (ABL) on the GPU. In each iteration all active blocks are updated and converged blocks are removed from the ABL. Neighbor blocks are also updated. Further, since points inside a data-group share the same CUDA block, they use shared memory simultaneously. Shared memory is on-chip memory at CPU register-speed and is much faster than global memory. In order to fully exploit fast shared memory access, we coalesced the memory by assigning points in the same data-group (a 2D block of 88 slice pixels) next to neighboring points in memory.

7.3 Result

All reported results use the Eikonal-SART reconstruction scheme presented above. All quality assessments are based on the global *L2* RMS error (ultrasound phantom vs. re-constructed densities). We created a physically-based human breast ultrasound phantom, based on the cryosection color images of the NIH Visible Female dataset ($http://www.nlm.nih.gov/research/visible/visible_human.html$). Hue was used to identify the basic tissues and their acoustical mapping was obtained by consulting biophysics tables (see Table 7). Lesions of varying size were also added. The phantom size is 128^2*40 and the smallest lesion is a sphere of a two-voxel diameter.

7.3.1 Reconstruction quality

We employed a high-quality wave equation solver to generate the projection data. Reconstructions (with 256 detectors per slice) were per-formed with the three numerical Eikonal equation solvers presented above (FMM, FSM, and FIM). All achieved nearly the same quality ($0.03 - 0.04$ RMSE). This is a vast improvement over a straight-ray reconstruction (0.25 RMSE) and demonstrates that accurate refraction modeling within a physically-based ray-tracing scheme enables reconstructions at high fidelity and accurate geometry. Fig. 32 gives visual evidence (for sound velocity SV) that the results are in excellent agreement with the original (acoustically mapped) phantom, with small detail being recovered well. Fig. 33 shows the attenuation reconstruction result (0.075 RMSE). Fig. 34 demonstrates that the 2D slice-by-slice reconstruction (right, RMSE 0.036) can reach nearly the same quality than a fully-3D reconstruction (center, RMSE 0.034). Here we observe that the smallest lesions (of two-voxel diameter close to the center) can be clearly identified in both reconstructions. Experiments with noisy data show that even with SNR= 10 (considered challenging) the inner lesions/structures remain very visible and distinguishable.

7.3.2 Time performance

With FMM we were able to reconstruct a 128^2 image from 256 transducers in about 60 seconds (3 iterations) on a $2.8GHz$ Pentium 4. For the $3D$ case, on a 128^3 grid, it takes 9.76 sec for a single wave propagation from one emitter. A full reconstruction with 256*128 transducers would then take about 533 hours. In contrast, the $2D$ slice decomposition approach would finish in 125 min, since the time per $2D$ wave tracking decreases to 0.03 sec. But this is still too slow for practical use.

Property	Tissue	Fat	Lesions (LG)	Lesions(SM)	Skin
Velocity	$1475m/s$	$1375m/s$	$1560m/s$	$1530m/s$	$1650m/s$
Attenuation	50	15	60/30	70	58

Table 7: Breast phantom ultrasound properties.

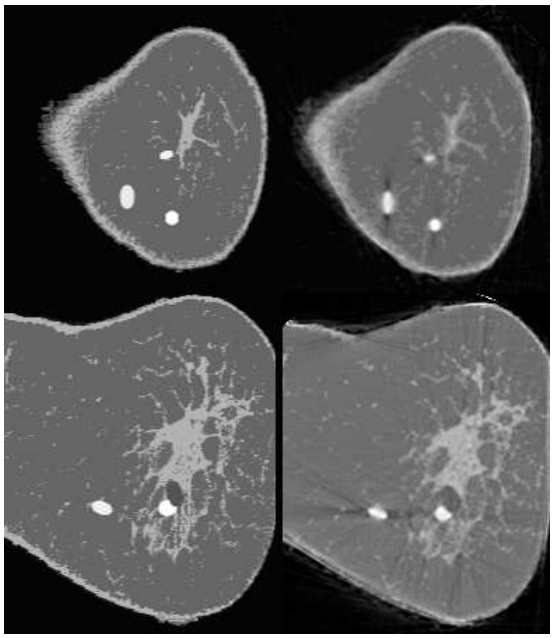


Figure 32: Speed image:(Left) acoustic breast phantom derived from the Visible Female dataset and (right) SV reconstructions. First row: slice close to the center, second row: slice close to the bottom.

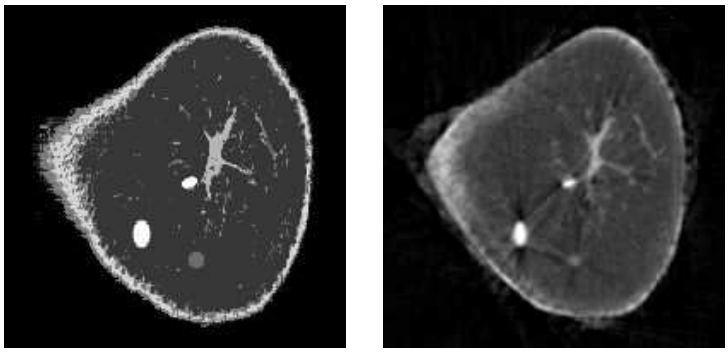


Figure 33: Attenuation image: (left) phantom, (right) reconstructed.

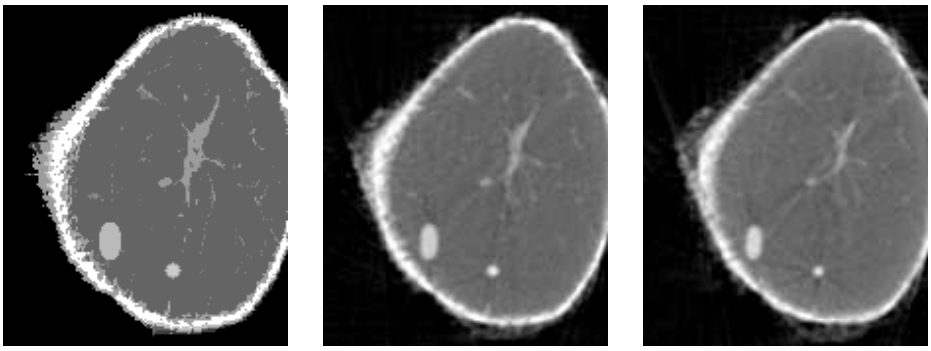


Figure 34: Fully 3D v.s. 2D slice-based reconstruction image: (left) phantom, (center) fully-3D, (right) 2D slice-based.

We then implemented FMM in CUDA, using an NVIDIA 8800 GTX GPU with 768 MB RAM and an NVIDIA Tesla with 1.5 *GB* RAM. To make full use of the GPU multiprocessors, it is essential to run a large number of concurrent threads. However, we were unable to beat CPU performance, even with a large number of concurrent threads. One reason is slow global memory access. Different threads do not share memory, and every thread requires frequent (and expensive) access to global memory. Another reason is that each thread kernel uses 28 registers (total is $8,192$) for the narrow band variables. This impeded the allocation of a sufficient number of threads, yielding a mere 0.33 multiprocessor occupancy. For the FSM, our CPU implementation took $0.031s$ $(0.18s)$ for one projection (wave front tracking) of a 128^2 (256^2) image. When run on a quad-core computer, a 3.5-fold speedup is achieved. On a distributed memory computer cluster with 8 nodes and 4 sub-domains and parallel sweep directions, we gained a 5-times speed up. Our FIM-implementation on CPU required $0.04s$ $(0.19s)$ for one projection of a 128^2 (256^2) image. The small-grain parallelism of the FIM favors an extension to multi-core, multi-processor, and GPU-SIMD configurations. The GPU implementation required only 0.002 sec for one projection of a 256^2 image, which is about 80 times faster than the fastest algorithm on the CPU (the FMM). Thus, for an object of size 256^2*44 and 512 transducers in each ring, it requires less than 5 minutes to reconstruct an object, which is a time that can easily satisfy real-life clinical scenarios.

Task	Grid size	FMM	FSM		FIM		
		CPU	CPU	Cluster-8 nodes	CPU	GPU (NVIDIA)	
						8086GTX	Tesla
Projection	1282	0.025	0.031	0.007	0.038	0.00082	0.00072
Projection	2562	0.097	0.180	0.039	0.189	0.0023	0.0022
Reconstr.	128240	2400	2649	506	3316	37.8	30.9
Reconstr.	256244	19200	32732	6393	37221	286.2	225.76

Table 8: Time (in s) required for one projection (wave propagation) for various Eikonal solvers and reconstructions.

The first two rows of Table 8 compare the time required for a single projection (wave front tracking) for two grid sizes and the second two rows compare the time required for $3D$ reconstruction using these three methods. We see that on the CPU, the one-pass approach of the FMM has the best performance, aided by the min-heap data structure. The FSM works well for multi-core and multiprocessor clusters (8 for projections and for reconstruction) since it can be computed in parallel, with little data communication. But clusters are expensive - an 8-node state-of-the-art computer cluster might cost thousands of US dollars. The GPU provides a good balance of price and performance. The FIM performs best on the GPU due its fine-grained parallelism. A NVIDIA Tesla costs only 600 U.S. dollars, yet it achieves about an 80-fold speedup over the fastest CPU FMM implementation.

7.4 Conclusions

We have demonstrated that iterative $3D$ UCT reconstruction with proper refraction physics, modeled directly in the object domain via wave front tracking can be accomplished at clinical rates. The reconstructions obtained from realistic phantoms resolve fine details well.

Chapter 8

Conclusions

To render volumetric datasets or traditional computer graphics objects like meshes accurately requires the simulation of materials. Current researches have improved largely in both accuracy and speed, but have mostly neglected a ubiquitous natural phenomenon, refraction. The material alteration in objects on the way of ray light can bend the light's direction, which leads to many cooperating components, including special high requirements on its filter and gradient filters, more burden on rendering due to the incoherence caused by direction changing, more time-consuming intersect computation when ray tracing is used, and more distortion and aliasing when reconstructing real medical imaging data. Obtaining high-quality refraction effects at reasonable rendering speeds has been so far a challenging task.

This thesis has demonstrated that in refraction gradient filters play an important factor in rendering quality, and the B-Spline filter achieves superior results, compared to the traditional Catmull-Rom filter, for the estimation of gradients from data. It does so at the same computational cost for the B-Spline-3, which has the same support, or at 3/4 of the cost for the B-Spline-2, which has a full spatial support of 3.0. We have also demonstrated that the B-spline allows one to balance smoothing with grid sample interpolation fidelity. This is beneficial in the presence of noise, round-off errors, and other artifacts incurred in the sampling of the original data.

In addition, we have implemented a refraction rendering system for a volumetric dataset, which introduced a variety of methods that allow high-quality interpolation mechanisms to be used without incurring the speed penalties that usually come with these. For this, we have described a novel octree-based ray acceleration method which specifically takes advantage of the nonnegativity property of B-spline kernels. These in turn produce results superior to the ones obtained with the more popular Catmull-Rom spline. Our approach for cell classification computes a cell mask that tightly fringes the refracting iso-surface, and consequently requires the testing of much fewer grid cells for iso-surface membership in the raycasting than previous strategies. Further, we have described a heuristic, yet effective method that defers super-sampling to a post-rendering process, achieving results of very similar rendering quality than traditional super-sampling, for a good variety of volume graphics objects.

Besides, for general graphics rendering, we proposed a novel framework for real-time inter reflection and refraction by ray tracing based on the geometry field. It combines a light field with a geometry image, and represents the intersection point of a surface with an arbitrary ray as a $4D$ lookup table. Conventional intersection testing in ray tracing is replaced by looking up in a geometry field, such that real time inter-surface reflection is achieved on current graphics hardware. Geometry field representation has several advantages in efficiency and simplicity. First, it reduces the memory cost from a common light field. In general, the storage requirement will be extremely high, if the position, normal and color material information are stored for each entry. Instead, only the texture coordinates are stored, such that the size of a geometry field is within the memory capacity of current hardware. Second, the geometry field can be represented as a generalized texture due to its regular structure, which fits to the design of common graphics hardware. Finally, the time cost is independent of the geometric complexity of the scene, and only dependent of the size of the geometry field. Application of the geometry field can be easily generalized for computing refractions and self reflection.

Further, for medical imaging data, we implemented a new method to correct the refraction in ultrasound computational tomography reconstruction. The key observation behind this method is that the up-winding Eikonal-equation-based

wave propagation method, Fast Marching Method (FMM), can determine the first-arriving phase in a continuous medium, which fits to the first-arrival signal measurement in ultrasound wave propagation. It also provides computational efficiency in determining the accurate paths of the non-linear rays caused by refraction distortion, which is ubiquitous phenomenon in breast ultrasound. We proposed two speed mechanisms for the FMM to trace the refractive rays: fixed speed and data driven relaxed speed. Our method is applicable in any UCT imaging situation in a moderately refractive media. UCT simulations, using a synthetic breast phantom, have demonstrated that the continuous wave propagation in HAFMM with data-driven relaxed speed achieves an accurate determination of the non-linear ray paths and provides much better fidelity in the image reconstruction. The reconstructed image quality improves by an order of magnitude compared to the pure straight ray method.

Finally, we have demonstrated that iterative 3D UCT reconstruction with proper refraction physics, modeled directly in the object domain via wave front tracking can be accomplished at clinical rates. FMM method replies on a heap data structure, which includes nodes from the entire wavefront. The computation for the next node of the wave front propagation at each step is sequential and globally dependent. We investigated the acceleration of three Eikonal solvers (Fast Marching Method (FMM), Fast Sweeping Method (FSM), Fast Iterative Method (FIM)) on three computational platforms (commodity graphics hardware (GPUs), multi-core CPUs, cluster computers), within our refractive Transmission Ultrasound CT framework. Our efforts provide insight into the capabilities of the various architectures for acoustic wave-front tracking, and they also yield a framework that meets the interactive demands of clinical practice, without a loss in reconstruction quality. The best algorithm for our accleration is FIM, which take full advantage of GPU acceleration. It maps well onto GPUs with parallel SIMD (Same Instruction Multiple Data) processors because of its fine-grained level parallelism, updating all AL points in parallel. In order to fully use the coherent memory access and control flow, our GPU implementation is based on a data-grouping scheme. Each point inside a data-group is executed as a thread, all threads for the same data-group in the AL combine into a block, and all blocks combine into a CUDA grid. Further,

since points inside a data-group share the same CUDA block, we use shared memory simultaneously, which is much faster than global memory. In order to fully exploit the fast shared memory access, we reorganized (coalesced) the memory by assigning points in the same data-group (we use a 2D block of 88 slice pixels or voxels) next to neighboring points in memory. A NVIDIA Tesla costs only 600 U. S. dollars, yet it achieves about an 80-fold speedup over the fastest CPU FMM implementation. The reconstructions obtained from realistic phantoms resolve fine details well.

Bibliography

[1] L. Amanatides. Ray tracing with cones. In *SIGGRAPH*, pages 129–135, 1984.

[2] A. H. Andersen and A. C. Kak. Simultaneous algebraic reconstruction technique (sart). In *Ultrason. Img.*, volume 6, pages 81–94, 1984.

[3] A. H. Anderson. A ray tracing approach to restoration and resolution enhancement in experimental ultrasound tomography. In *Ultrason. Img.*, volume 12, pages 268–291, 1990.

[4] J. A. Barentzen. On the implementation of fast marching methods for 3d lattices. In *Tech. report, Tech. Univ. of Denmark*, 2001.

[5] M. J. Bentum, B. B. A. Lichtenbelt, and T. Malzbender. Frequency analysis of gradient estimators in volume rendering. In *IEEE Trans. Vis. Comput. Graph.*, volume 2(3), pages 242–254, 1996.

[6] J. F. Blinn. Simulation of wrinkled surfaces. In *Computer Graphics*, volume 12(3), pages 286–292, 1978.

[7] J. F. Blinn and M. E. Newell. Texture and reflection in computer generated images. *Commun. ACM*, 19(10):542–547, 1976.

[8] K. Bliznakova, Z. Bliznakov, V. Bravou, Z. Kolitsi, and N. Pallikarakis. A three-dimensional breast software phantom for mammography simulation. In *Phys. Med. Biol.*, volume 48, pages 3699–3719, 2003.

[9] M. Boue and P. Dupuis. Markov chain approximations for deterministic control problems with affine dynamics and quadratic costs in the control. In *SIAM J. Numerical Analysis*, volume 36, pages 667–695, 1999.

[10] R. P. Brent. *Algorithms for Minimization Without Derivatives*. Prentice-Hall, 1973.

[11] N. A. Carr, J. D. Hall, and J. C. Hart. The ray engine. In *ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 37–46, 2002.

[12] W.-C. Chen, J.-Y. Bouguet, M. H. Chu, and R. Grzeszczuk. Light field mapping: efficient representation and hardware rendering of surface light fields. In *SIGGRAPH*, pages 447–456, 2002.

[13] S. Davis and C. Wyman. Interactive refractions with total internal reflections. In *Proceedings of Graphics Interface*, 2007.

[14] F. Denis, O. Basset, and G. Gimenez. Ultrasonic transmission tomography in refracting media: Reduction of refraction artifacts by curved-ray techniques. In *IEEE Trans. Med. Img.*, volume 14, pages 173–188, 1995.

[15] P. J. Diefenbach and N. I. Badler. Multi-pass pipeline rendering: realism for dynamic environments. In *Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 59–ff., 1997.

[16] E. W. Dijkstra. A note on two problems in connection with graphs. In *Numeric Mathematics*, volume 1, pages 269–271, 1959.

[17] N. Duric and P. Littrup. Detection of breast cancer with ultrasound tomography: First results with the computerized ultrasound risk evaluation (c.u.r.e) prototype. In *Medical Physics*, volume 34(2), pages 733–785, 2007.

[18] N. Duric, P. Littrup, A. Babkin, D. Chambers, S. Azevedo, R. Pevzner, M. Tokarev, E. Holsapple, O. Rama, and R. Duncan. Development of ultrasound tomography for breast imaging: Technical assessment. In *Am. Assoc. Phys. Med.*, volume 32(5), pages 1375–1386, 2005.

[19] T. Foley and J. Sugerman. Kd-tree acceleration structures for a gpu raytracer. In *ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 15–22, 2005.

[20] S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen. The lumigraph. In *SIGGRAPH*, pages 43–54, 1996.

[21] N. Greene. Environment mapping and other applications of world projections. *IEEE Computer Graphics and Applications*, 6(11):21–29, 1986.

[22] E. Grller. Nonlinear ray tracing: Visualizing strange worlds. In *The Visual Computer*, volume 11(5), pages 263–274, 1995.

[23] X. Gu, S. J. Gortler, and H. Hoppe. Geometry images. In *SIGGRAPH*, pages 355–361, 2002.

[24] X. Gu and S.-T. Yau. Global conformal parameterization. In *Symposium on Geometry Processing*, pages 127–137, 2003.

[25] S. Guy and C. Soler. Graphics gems revisited: fast and physically-based rendering of gemstones. In *ACM Trans. Graph.*, volume 23(3), pages 231–238, 2004.

[26] Z. S. Hakura and J. M. Snyder. Realistic reflections and refractions on graphics hardware with hybrid rendering and layered environment maps. In *Proceedings of the 12th Eurographics Workshop on Rendering Techniques*, pages 289–300, 2001.

[27] P. S. Heckbert and P. Hanrahan. Beam tracing polygonal objects. In *Computer Graphics*, volume 18(3), 1984.

[28] W. Heidrich, H. P. A. Lensch, M. F. Cohen, and H.-P. Seidel. Light field techniques for reflections and refractions. In *Rendering Techniques*, pages 187–196, 1999.

[29] http://www.povray.org.

[30] W. Hu and K. Qin. Interactive approximate rendering of reflections, refractions, and caustics. In *IEEE Transactions on Visualization and Computer Graphics*, pages 319–329, 2006.

[31] V. Interrante, H. Fuchs, and S. M. Pizer. Conveying the 3d shape of smoothly curving transparent surfaces via texture. In *IEEE Trans. Vis. Comput. Graph.*, volume 3(2), pages 98–117, 1997.

[32] W. Jeong, P. Fletcher, R. Tao, and R. Whitaker. Interactive visualization of volumetric white matter connectivity in dt-mri using a parallel-hardware hamilton-jacobi solver. In *IEEE Trans. Visualization and Computer Graphics*, volume 13(6), pages 1480–1487, 2007.

[33] A. J. D. K. T. Ladas. Application of an art in an experimental study of ultrasonic diffraction tomography. In *Ultrason. Img.*, volume 15, pages 48–58, 1993.

[34] T. L. Kay and J. T. Kaijia. Ray tracing complex scenes. In *Computer Graphics*, volume 20, pages 269–278, 1986.

[35] G. L. Kindlmann and J. W. Durkin. Semi-automatic generation of transfer functions for direct volume rendering. In *Volvis*, 1998.

[36] J. Kniss, G. L. Kindlmann, and C. D. Hansen. Multidimensional transfer functions for interactive volume rendering. In *IEEE Trans. Vis. Comput. Graph.*, volume 8(3), pages 270–285, 2002.

[37] J. Kruger, K. Burger, and RudigerWestermann. Interactive screenspace accurate photon tracing on gpus. In *In Rendering Techniques (Eurographics Symposium on Rendering - EGSR)*, pages 319–329, 2006.

[38] M. Levoy and P. Hanrahan. Light field rendering. In *SIGGRAPH*, pages 31–42, 1996.

[39] A. Li and K. Mueller. Methods for efficient, high quality volume resampling in the frequency domain. In *IEEE Visualization*, pages 3–10, 2004.

[40] S. Li and K. Mueller. Accelerated high quality refraction computations for volume graphics. In *International Workshop on Volume Graphics*, 2005.

[41] S. Li, K. Mueller, M. Jackowski, D. Dione, and L. Staib. Fast marching method to correct for refraction in ultrasound computed tomography. In *IEEE Symp.Biomed. Img.*, pages 896–899, 2006.

[42] D. Lischinski and A. Rappoport. Image-based rendering for non-diffuse synthetic scenes. In *Rendering Techniques*, pages 301–314, 1998.

[43] F. Losasso, H. Hoppe, S. Schaefer, and J. D. Warren. Smooth geometry images. In *Symposium on Geometry Processing*, pages 138–145, 2003.

[44] S. R. Marschner and R. Lobb. An evaluation of reconstruction filters for volume rendering. In *IEEE Visualization*, pages 100–107, 1994.

[45] V. Masselus, P. Peers, P. Dutré, and Y. D. Willems. Relighting with 4d incident light fields. *ACM Trans. Graph.*, 22(3):613–620, 2003.

[46] C. R. Meyer, T. L. Chenevert, and P. L. Carson. A method for reducing multipath artifacts in ultrasonic computed tomography. In *J. Acoust. Soc. Am.*, volume 72(3), pages 820–823, 1982.

[47] T. Mller, R. Machiraju, K. Mueller, and R. Yagel. Evaluation and design of filters using a taylor series expansion. In *IEEE Trans. Vis. Comput. Graph.*, volume 3(2), pages 184–199, 1997.

[48] S. J. Norton. Computing ray trajectories between two points: A solution to the ray-linking problem. In *Optical Soc. of America*, volume 4(10), pages 1919–1922, 1987.

[49] K. M. Pan and C. N. Liu. Tomographic reconstruction of ultrasonic attenuation with correction for refractive errors. In *IBM J. Res. Develop.*, volume 25(1), pages 71–82, 1981.

[50] X. Pan. Unified reconstruction theory for diffraction tomography with consideration of noise control. In *J. Opt. Soc. Am. A*, volume 15, pages 2312–2326, 1998.

[51] S. G. Parker, M. Parker, Y. Livnat, P. P. J.Sloan, C. D. Hansen, and P. Shirley. Interactive ray tracing for volume visualization. In *IEEE Trans. Vis. Comput. Graph.*, volume 5(3), pages 238–250, 1999.

[52] T. Purcell, I. Buck, W. Mark, and P. Hanrahan. Ray tracing on programmable graphics hardware. In *ACM Trans. Graphics*, volume 21(3), pages 703–712, 2002.

[53] T. J. Purcell, I. Buck, W. R. Mark, and P. Hanrahan. Ray tracing on programmable graphics hardware. In *SIGGRAPH*, pages 703–712, 2002.

[54] B. D. S. Q. Zhu. Wavefront amplitude distribution in the female breast. In *J. Acous. Soc. of America*, volume 96(1), pages 1–9, 1996.

[55] Y. Quan and L. Huang. Sound-speed tomography using first-arrival transmission ultrasound for a ring array. In *Proc. SPIE Medical Imaging*, page 6513, 2007.

[56] D. Rodgman and M. Chen. Refraction in discrete ray tracing. In *Volume Graphics*, pages 3–17, 2001.

[57] D. Rodgman and M. Chen. *Volume Denoising for Visualizing Refraction*. Springer, 2004.

[58] P. V. Sander, Z. J. Wood, S. J. Gortler, J. Snyder, and H. Hoppe. Multi-chart geometry images. In *Symposium on Geometry Processing*, pages 146–155, 2003.

[59] C. M. Schmidt. Simulating refraction using geometric transforms. In *Master's thesis, Computer Science Department, University of Utah*, 2003.

[60] J. Sethian and A. Vladimirsky. Ordered upwind methods for static hamilton-jacobi equa-tions: Theory and algorithms. In *SIAM J. of Numerical Analysis*, volume 41(1), pages 325–363, 2003.

[61] J. A. Sethian. Level set methods and fast marching methods: Evolving interfaces in geometry, fluid mechanics, computer vision, and material science. In *Cambridge Monographs on Applied and Computational Mathematics*, 1999.

[62] L. Szirmay-Kalos, B. Aszódi, I. Lazányi, and M. Premecz. Approximate ray-tracing on the gpu with distance impostors. In *Eurographics*, 2005.

[63] L. Szirmay-Kalos, B. Aszodi, I. Lazanyi, and M. Premecz. Approximate ray-tracing on the gpu with distance impostors. In *Computer Graphics Forum*, volume 24(3), pages 695–704, 2005.

[64] P. Taylor, R. Owens, and D. Ingram. Simulated mammography using synthetic 3d breasts digital mammography. In *Proc. 4th Int. Workshop on Digital Mammography*, pages 283–290, 1998.

[65] P. Thevenaz, T. Blu, and M. Unser. Interpolation revisited. In *IEEE Trans. Med. Img.*, volume 19, pages 739–758, 2000.

[66] P. Thevenaz and M. Unser. Precision isosurface rendering of 3-d image data. In *IEEE Trans. Img. Proc.*, volume 12(7), 2003.

[67] G. E. Trahey, P. D. Freiburger, L. F. Nock, and D. C. Sullivan. In vivo measurement of ultrasonic beam distortion in the breast. In *Ultrason. Img.*, volume 13, pages 71–90, 1999.

[68] J. Tsitsiklis. Efficient algorithms for globally optimal trajectories. In *IEEE Trans. on Automatic Control*, volume 40(9), pages 1528–1538, 1995.

[69] P. Y. Ts'o and B. A. Barsky. Modeling and rendering waves: wave-tracing using beta-splines and reflective and refractive texture mapping. In *ACM Trans. Graph.*, volume 6(3), pages 191–214, 1987.

[70] M. Unser. Splines - a perfect fit for signal and image processing. In *IEEE Signal Processing Magazine*, 1999.

[71] M. Unser, A. Aldroubi, and M. Eden. B-spline signal processing: Part i-theory. In *IEEE Trans. on Sig. Proc.*, volume 41(2), pages 821–832, 1993.

[72] M. Unser, A. Aldroubi, and M. Eden. B-spline signal processing: Part ii - efficient design and applications. In *IEEE Trans. on Sign. Proc.*, volume 41(2), pages 834–848, 1993.

[73] I. Wald, T. Kollig, C. Benthin, A. Keller, and P. Slusallek. Interactive global illumination using fast ray tracing. In *In Proceedings of the Eurographics Rendering Workshop*, pages 15–24, 2002.

[74] D. Weiskopf, T. Schafhitzel, and T. Ertl. Gpu based nonlinear ray tracing. 23(3):625–634, 2004.

[75] T. Whitted. An improved illumination model for shaded display. In *Commun. ACM*, volume 23(6), pages 343–349, 1980.

[76] D. N. Wood, D. I. Azuma, K. Aldinger, B. Curless, T. Duchamp, D. Salesin, and W. Stuetzle. Surface light fields for 3d photography. In *SIGGRAPH*, pages 287–296, 2000.

[77] S. Woop, J. Schmittler, and P. Slusallek. Rpu: a programmable ray processing unit for realtime ray tracing. *ACM Trans. Graph.*, 24(3):434–444, 2005.

[78] C. Wyman. An approximate image-space approach for interactive refraction. In *ACM Transactions on Graphics*, volume 24(3), pages 1050–1053, 2005.

[79] C. Wyman. Interactive image-space refraction of nearby geometry. In *Proceedings of GRAPHITE*, pages 205–211, 2005.

[80] L. Yatziv, A. Bartesaghi, and G. Sapiro. O(n) implementation of the fast marching algo-rithm. In *Journal of Computational Physics*, volume 212, pages 393–399, 2006.

[81] J. Yu, J. Yang, and L. McMillan. Real-time reflection mapping with parallax. In *ACM SIGGRAPH 2005 Symposiumon Interactive 3D Graphics and Games*, 2005.

[82] H. Zhao. Fast sweeping method for eikonal equations. In *Mathematics of Computation*, volume 74, pages 603–627, 2005.