Volume Graphics (2005)
E. Gröller, I. Fujishiro (Editors)

Accelerated, High-Quality Refraction Computations for Volume Graphics

Shengying Li and Klaus Mueller

Department of Computer Science, Center for Visual Computing, Stony Brook University

Abstract

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 coefficients, 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 with pre-classified cells that is carefully matched to the filter and voxel neighborhood characteristics. Here, we benefit greatly from the non-negativity of the B-Spline kernel. Finally, we describe an innovative scheme that achieves the high quality of pixel super-sampling on a flat backdrop plane without the overhead of tracing the actual rays across the refractive object.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism, I.3.3 [Computer Graphics-Picture/Image Generation] Anti-aliasing, G.1.1 [Numerical Analysis]: Interpolation

1 Introduction

The rendering of refractive objects has fascinated researchers from the early days of computer graphics [KK86][Whi80]. 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 POV-Ray [POV] 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 [RC01] 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 datasets, 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, 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. This is a prime reason why the quality of the filter used for interpolation and gradient (refractive index) calculation is so important, and a related paper [LM05] has addressed these filter aspects in great detail. In that prior 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, and Section 3.1 will summarize these insights in order to justify some of the choices we have made for the work reported here.

In this current paper, 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 discretely sampled 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 [GS04], which employed Heckbert's beam-tracing [HH84] approach to manage the rays refracted at the polygonal surface that modeled the many facets of the diamond. However, in contrast to that work, 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, which is the common data representation in volume graphics.

email: {lshengyi, mueller}@cs.sunysb.edu

In volume graphics, the degree of object smoothness is determined by the interpolation filter, and the level of available detail is determined by the resolution of the grid. The former poses certain demands on the filters used for object reconstruction, while the latter is bounded by data storage and management constraints. Another key issue is the rendering speed, which is directly affected by the number of grid interpolation calculations that need to be performed. In this paper, we offer an elegant method based on a filtered octree with pre-classified cells to identify a refractive iso-surface quickly, while still employing high-quality interpolation filters. These two methods combined allow a near-interactive design of refractive volume graphics objects with high-quality visual feedback. These acceleration methods are applicable for any

Since our focus is the volume graphics object and the evaluation and design of its refractive properties, we prefer to employ a simple shape, such as a plane, to hold the background texture used for illustrating the resulting distortion patterns. This eliminates possibly confounding visual effects that might otherwise arise from a competing shape complexity of the background object. In addition, we shall see that this restriction to planar backdrop images also allows for an attractive acceleration for anti-aliasing.

The structure of our paper is as follows. In Section 2 we present related work in this area, while Section 3 addresses the theoretical aspects of the current work and outlines present problems. Section 4 describes our actual implementation, and Section 5 and 6 present results and conclusions, respectively.

2 Related Work

Refraction has been the subject of much interest even early on in computer graphics, where Kay [KK86] was the first to apply Snell's law to model this optical effect. Soon after, Whitted [Whi80] produced the now famous image of floating transparent balls over a checkerboard to show off his recursive ray tracing framework. Amanatides [Ama84] 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 [RC01], 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 in that context, they mention the difficulties associated with correctly separating two subdomains to estimate the refraction index. In another paper

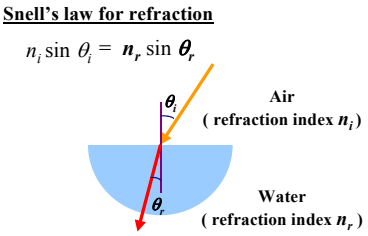


Figure 1: *Snell's law for refraction*.

[RC04], 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 Gröller et al. [Grö95], 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. [WSE04] 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. In our work, we are more interested in the design of surface-based volumetric objects and thus our filtered octree-based space decomposition method to accelerate the rays is more appropriate.

3 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:

$$n_i \sin\theta_i = n_r \sin\theta_r$$

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 Fig. 1).

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 a previous paper [LM05], 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.

3.1 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 [LM05] 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 [UAE93], this is just a one-time 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 [UAE93][UAE93]. 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.

3.2 Super-sampling

Part of the aliasing stems from sampling the background texture at a rate lower than the Nyquist frequency requirement. Fig. 2 shows the effect of under-sampling, when illuminating a simple glass sphere with refracted light rays passing through. Here we observe, that at the two ends of the line in this figure the rays bend at much larger angle than the rays around 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 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.

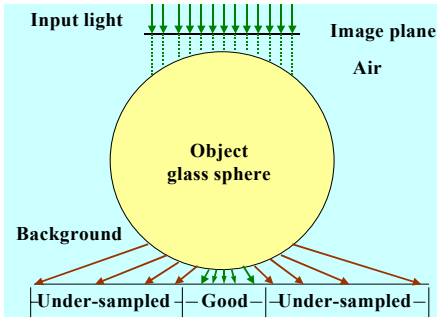


Figure 2: *Illustration of object-dependent backplane sampling.*

3.3 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 [PPL*99]. For the higher-order filters, such as cubic convolution and B-Spline, we use Brent's iterative root finding method [Bre73] since at these higher function orders no analytic root can be found. Brent's method works efficiently for B-splines due to the non-negativity property of the B-spline basis function.

4 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.

4.1 DDA and octree

The 3D-DDA algorithm, such as Bresenham's 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 3D-DDA 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 [RC01] and therefore provides for faster rendering.

However, the speed can still be improved by space-decomposition 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.

4.2 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 [TU03] or, better, by a novel scheme that exploits the monotonic-decay and non-negativity properties of the B-Spline filter. Both of these methods are described next.

4.2.1. Binary morphologic cell-classification (BMCC)

In [TU03] a method using binary morphology in the pre-processing 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 neighbor*s (see Fig. 3).

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.

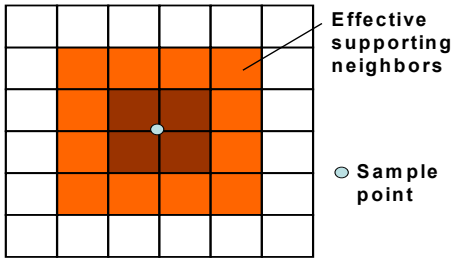


Figure 3: All colored cells contain effective supporting neighbor points for the sample point shown, here for a B-Spline of order 2. The darker the color is, the larger the support.

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.

4.2.2. Continuous cell-classification (CCC)

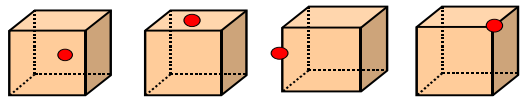
Determining the maximum and minimum possible value inside a cell is an optimization problem for maximizing and minimizing the interpolating function:

$$f(x) = \sum_k c_k \phi(x-k) \text{, where } i \le x \le i+1 \text{, } k, i \in Z^q .$$

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 Fig. 4 (next page). 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



(a) (b) (c) (d)

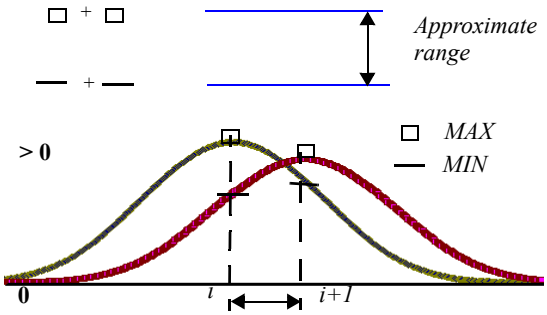
Figure 4: Cases for critical point with maximum or minimum value in a cell, (a) inside a cell, (b) on a face, (c) on an edge, (d) on a vertex.

solve. We therefore decided to employ an approximate method, as is described next.

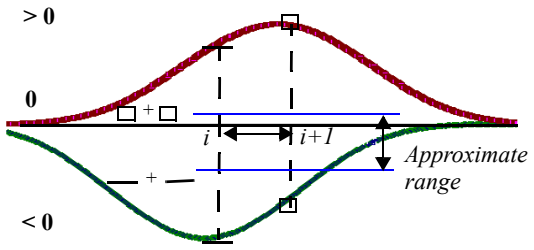
4.2.3. Approximate CCC (ACCC)

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.

Fig. 5 sketches the idea in 1D. 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



(a) both grid points positive



(b) one grid point is negative

Figure 5: *Sketch of the ACCC method: For any point between grid point* i *and* i+1, *its value is always within the range* Sum(MIN_j) *and* Sum(MAX_j), *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).

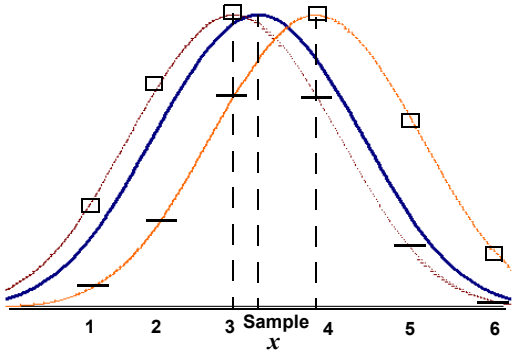


Figure 6: *Illustration of the ACCC method with two grid point kernels: The kernel for any sample point between grid point 3 and 4 is within the narrow band of the kernels on grid point of 3 and 4.*

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).

More formally (see also Fig. 6), when the absolute distance between a grid point and an arbitrary sample point $\boldsymbol{x}$ is within i and $(i+1)$ $(i \in Z,\ i > 0)$,the interpolated value at $\boldsymbol{x}$ is:

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

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

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

$$Max(f(x)) = f_{iso} + Max\left(\sum_k (c_k - f_{iso})\phi(x-k)\right)$$

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

where

$$Max\phi(x-k) = \phi(i), \quad where(i \leq |x-k| < i+1)$$

$$Min\phi(x-k) = \phi(i+1), \quad where(i \leq |x-k| < i+1)$$

$$k, i \in Z^q$$

Likewise, the lower bound of the interpolation value for $\boldsymbol{x}$ is given by:

$$Min(f(x)) = f_{iso} + Min\left(\sum_k (c_k - f_{iso})\phi(x-k)\right)$$

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

Therefore, by computing the lower and upper bound of the interpolation value that $\boldsymbol{x}$, 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:

$$f(x) = \sum_{i,j,k} c_{ijk}\phi(x-k)\phi(x-j)\phi(x-k)$$
$$= \sum_{i,j,k} (c_{ijk}-f_{iso})\phi(x-i)\phi(x-j)\phi(x-k)+f_{iso}$$

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

4.2.4 Discussion

In [TU03], 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 look-up 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 $64 \times 64 \times 64$, 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 B-Spline), 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 [TU03], this progressive refinement strategy would trade computational efficiency for higher accuracy.

4.3 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.

Our PRSS method makes use of the space coherence of a matrix of neighboring rays. Once we obtain each ray's 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. Fig. 7 (next page) explains the individual steps of our 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 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 Fig. 7) 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 inter-

$$Max(f(x)) = f_{iso} + Max\left(\sum_{i,j,k}(c_{ijk}-f_{iso})\phi(x-i)\phi(x-j)\phi(x-k)\right)$$
$$= f_{iso} + \sum_{i,j,k}(c_{ijk}-f_{iso})\begin{cases} Max\phi(x-i)Max\phi(x-j)Max\phi(x-k), & if((c_{ijk}-f_{iso})\geq 0) \\ Min\phi(x-k)Min\phi(x-j)Min\phi(x-k), & if((c_{ijk}-f_{iso})<0) \end{cases} \quad (1)$$

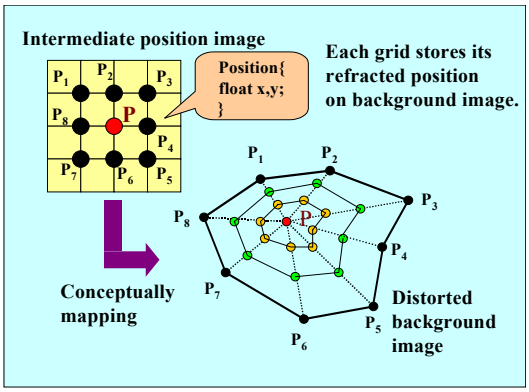


Figure 7: *Post-refraction super-sampling*

mediate 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.

5 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.

5.1 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.

Fig. 8 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 SS) and without super-sampling. We observe that without supersampling some jaggies along the edges of the distorted checkerboard remain, especially in the sphere's 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 Fig. 8 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 non-supersampled 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).

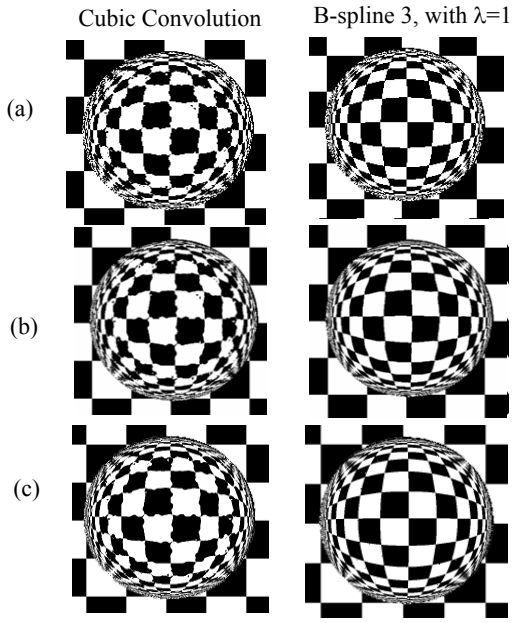


Figure 8: *Refracted smooth sphere comparing Catmull-Rom Cubic and B-spline 3 with smooth λ=1 in three different categories. (a) without using super-sampling. (b) Traditional super-sampling. (c) Post-refraction super-sampling.*

	Without SS	Real SS	PRSS
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 Fig. 8, rendered with raytracing at a stepsize of 0.1.*

	raycast	BMCC		ACCC	
	step=0.1	DDA	Octree	DDA	Octree
Sphere	6.65	0.84	0.29	0.69	0.15
cthead	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. On average, DDA increases the speed around 7 times comparing to raycast, while the octree improves DDA about 3 times. The ACCC saves about 50% time over BMCC.*

5.2 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 and ACCC, using PRSS and B-spline 3, as well as Brent's 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 B-Splines, 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.

Data	Size	Spline2	Spline3
Sphere	64 × 64 × 64	67.1%	37.5%
ctHead	128 × 128 × 128	75.2%	51.3%
lobster	320 × 320 × 34	62.3%	60.6%
teapot	256 × 256 × 178	77.1%	51.4%

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

5.3 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 Fig. 9 and Fig. 10). The resulting images also illustrate the complexity that can result once there are chains of refractive surfaces. For example, in Fig. 10 (b) and (c), the small spheres are originally in opposite positions inside the big sphere, while the refraction image places them both in the lower-left 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, Fig. 11 represents the refraction results obtained with a volumetric teapot, while Fig. 12 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 Fig. 13, now using a photograph as the background image to act as an imposter for a real scene.

	DDA					Octree					Save
	Linear	C.C.	Spline 2	Spline 3	Spline 6	Linear	C.C.	Spline 2	Spline 3	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%
cthead	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.*

6 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 non-negativity property of B-spline kernels. These in turn produce results superior to the ones obtained with the more popular Catmull-Rom spline (which also does have this property) (see also [LM05]). 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.

Acknowledgments

This research was partially supported by NSF CAREER grant ACI-0093157.

References

[Ama84] J. Amanatides: Ray tracing with cones. In *Computer Graphics (SIGGRAPH'84),* 18(3), 129-135, 1984.

[BLM96] M. Bentum, B. Lichtenbelt, T. Malzbender: Frequency analysis of gradient estimators in volume rendering. *IEEE Trans. Visualization & Computer Graphics,* 2(3): 242-254, 1996.

[Bli78] J. Blinn: Simulation of wrinkled surfaces. *Computer Graphics,* 12(3):286-292, 1978.

[Bre73] R. Brent, Ed: *Algorithms for Minimization Without Derivatives*, Prentice-Hall, 1973.

[Grö95] E. Gröller: Nonlinear ray tracing, visualizing strange worlds. *The Visual Computer,* 11(5): 263-274, 1995.

[GS04] S. Guy, C. Soler: Graphics gems revisited. *ACM Trans. on Graphics (SIGGRAPH'04)*, pp. 231-238, 2004.

[HH84] P. Heckbert, P. Hanrahan: Beam tracing polygonal objects. *Computer Graphics (SIGGRAPH'84)*, 18(3), pp. 119-127, 1984.

[IFP97] V. Interrante, H. Fuchs, S. Pizer: Conveying the 3D shape of smoothly curving transparent surfaces via texture. *IEEE Trans. Visualization & Computer Graph*ics, 3(2): 98-117, 1997.

[KK86] T. Kay, J. Kaijia: Ray tracing complex scenes. *Computer Graphics (SIGGRAPH'86)*, 20(3), 269-278, 1986.

[KD98] G. Kindlmann, J. Durkin: Semi-Automatic generation of transfer functions for direct volume rendering, *Volume Rendering Symposium'98*, pp. 79-86, Chapel Hill, October 1998.

[KKH02] J. Kniss, G. Kindlmann, C. Hansen: Multidimensional transfer functions for interactive volume rendering. *IEEE Trans. Visualization & Computer Graphics,* 8(3): 270-285, 2002.

[LM04] A. Li, K. Mueller: Methods for efficient, high quality volume resampling in the frequency domain. *IEEE Visualization'04*, pp. 3-10, Austin, October 2004.

[LM05] S. Li, K. Mueller: Spline-based gradient filters for high-quality refraction computation in discrete datasets. *EuroVis 2005*, Leeds, June 2005.

[ML94] S. Marschner, R. Lobb: An evaluation of reconstruction filters for volume rendering. *IEEE Visualization'94*, pp. 100-107, 1994.

[Mae88] E. Maeland: On the comparison of interpolation methods. *IEEE Trans. on Medical Imaging,* 7(3), 213-217, 1988.

[MMM*97] T. Möller, R. Machiraju, K. Mueller, R. Yagel: Evaluation and design of filters using a Taylor Series expansion. *IEEE Trans. on Visualization & Computer Graph*ics, 3(2): 184-199, 1997.

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

[PPL*99] S. G. Parker, M. Parker, Y. Livnat, P. Pike J. Sloan, C. D. Hansen, P. Shirley: Interactive ray tracing for volume visualization. *IEEE Trans. on Visualization & Computer Graph*ics 5(3): 238-250, 1999.

[RC01] D. Rodgman, M. Chen: Refraction in discrete ray tracing. *Volume Graphics Workshop 2001,* pp. 3-17, Stony Brook, June 2001.

[RC04] D. Rodgman, M. Chen: Volume Denoising for Visualizing Refraction. *Proc. Dagstuhl Scientific Visualization 2003, Mathematics and Visualization Series*, Springer, 2004.

[TU03] P. Thevenaz, M. Unser: Precision isosurface rendering of 3-D image data. *IEEE Trans. Image Processing*, 12(7): 764-775, 2003.

[TBU00] P. Thevenaz, T. Blu, M. Unser: Interpolation revisited. *IEEE Trans. Medical Imaging*, 19(7): 739-758, 2000.

[Uns99] M. Unser: Splines - A perfect fit for signal and image processing. *IEEE Signal Processing Magazine,* 16(6): 22-38, 1999.

[UAE93] M. Unser, A. Aldroubi, M. Eden: B-spline signal processing: Part I-Theory. *IEEE Trans. Signal Processing*, 41(2): 821-832, 1993.

[UAE93] M. Unser, A. Aldroubi, M. Eden: B-spline signal processing: Part II - Efficient design and applications. *IEEE Trans. Signal Processing*, 41(2): 834-848, 1993.

[WSE04] D. Weiskopf, T. Schafhitzel, T. Ertl: GPU-based nonlinear ray tracing. *Computer Graphics Forum,* 23(3): 625-634, 2004.

[Whi80] T. Whitted: An improved illumination model for shaded display. *Communications ACM* 23(6): 343-349, 1980.

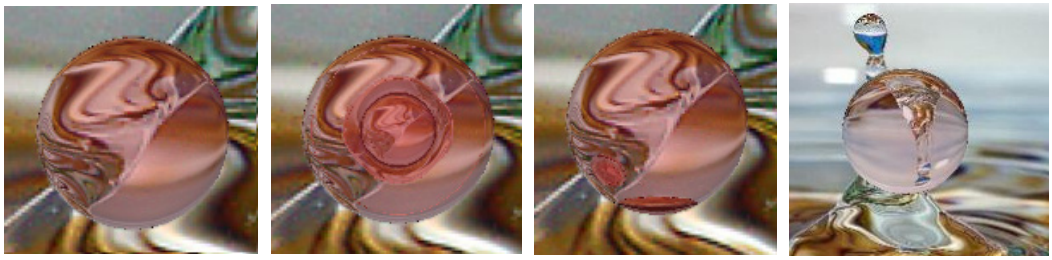


Figure 9: *Various volume graphics renderings of refractive glass balls: (center) the ball with one and two air holes, respectively; (left and right) the ball with no air holes and different backdrops.*

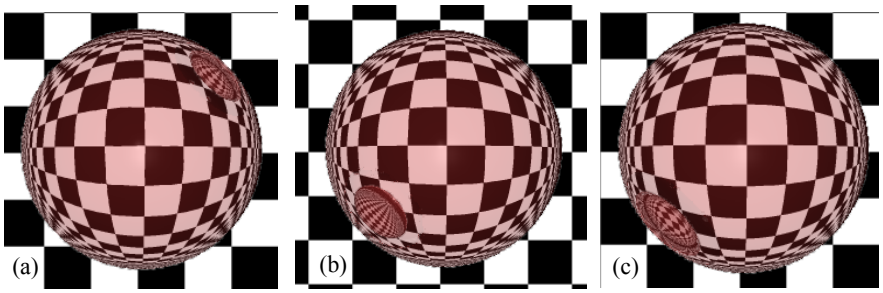
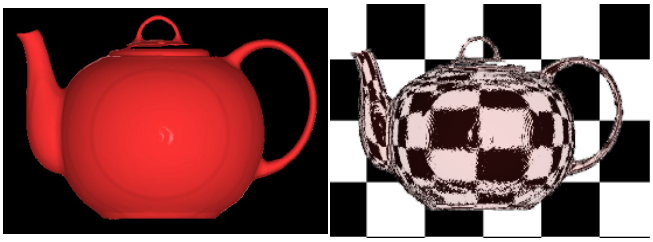
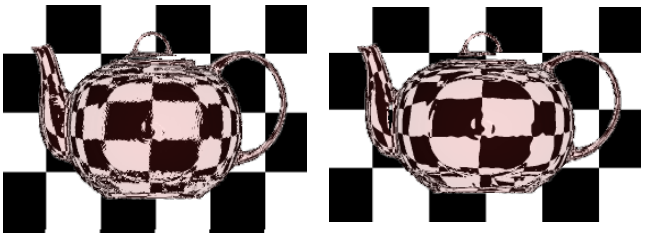


Figure 10: *Multiple layers of refraction in a sphere.*

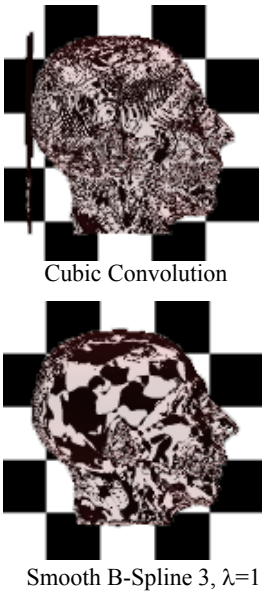


Rendered Result Cubic Convolution



B-Spline 2 without prefiltering Smooth B-Spline 3, λ=1

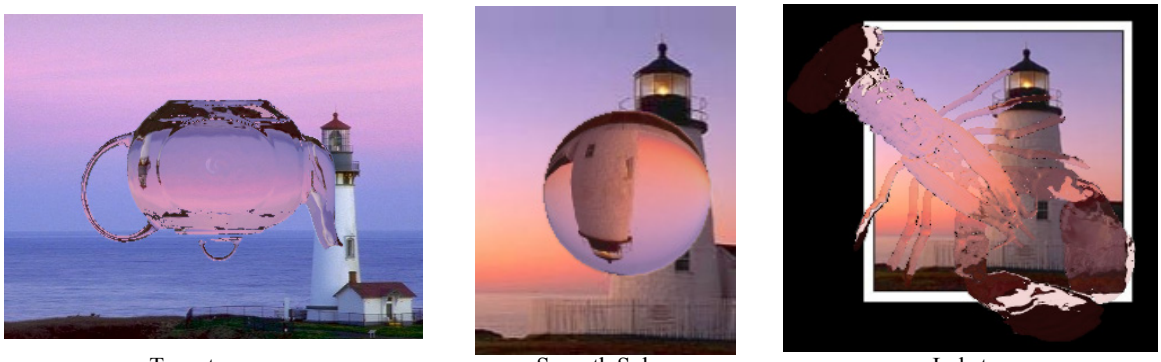
Figure 11: *Refracted results of teapot.*



Cubic Convolution

Smooth B-Spline 3, λ=1

Figure 12: *Refractive rendering of a CT-head.*



Teapot Smooth Sphere Lobster

Figure 13: *Refraction images of teapot, smooth sphere, and lobster using a Smooth-B-Spline 3.*