The Magic Volume Lens: An Interactive Focus+Context Technique for Volume Rendering

Lujin Wang   Ye Zhao   Klaus Mueller   Arie Kaufman*

Center for Visual Computing, Computer Science, Stony Brook University

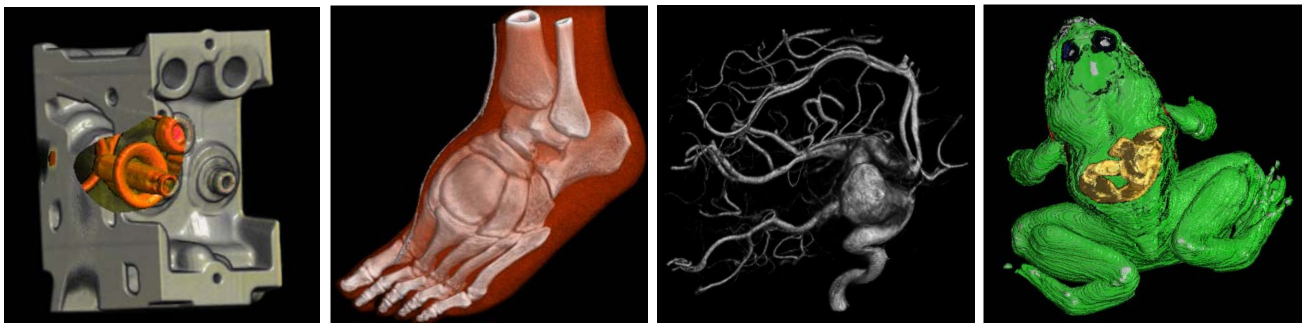

(a) (b) (c) (d)

Figure 1: Magic volume lens results. (a) magnifying inside features in an arbitrary-shaped area on an engine, (b) applying sampling-rate-based lens on a foot, (c) enlarging area of interest on an aneurism, (d) magnifying the duodenum of a segmented frog dataset.

Abstract

The size and resolution of volume datasets in science and medicine are increasing at a rate much greater than the resolution of the screens used to view them. This limits the amount of data that can be viewed simultaneously, potentially leading to a loss of overall context of the data when the user views or zooms into a particular area of interest. We propose a focus+context framework that uses various standard and advanced magnification lens rendering techniques to magnify the features of interest, while compressing the remaining volume regions without clipping them away completely. Some of these lenses can be interactively configured by the user to specify the desired magnification patterns, while others are feature-adaptive. All our lenses are accelerated on the GPU. They allow the user to interactively manage the available screen area, dedicating more area to the more resolution-important features.

CR Categories: I.3.3 [Computer Graphics]: Picture/Image Generation—Viewing algorithms; I.3.3 [Computer Graphics]: Picture/Image Generation—Display algorithms I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Color, shading, shadowing, and texture

Keywords: Focus+Context Techniques, Lens, Volume Rendering, Hardware-assisted Volume Rendering

1 Introduction

Recent years have seen a dramatic growth in our ability to compute, acquire, and assemble datasets of increasingly large magnitudes and resolutions. Great advances have also been made in screen technology, bringing high-resolution displays to the desktop at affordable prices, as well as offering sophisticated CAVE environments. The one device that has consistently resisted participation in this spiral of growth is the human eye and the cortical visual processing abilities. In fact, there is a natural limit on the screen pixel density, as a function of distance, which the human eye can resolve, and there is also a natural falloff of retinal receptor density towards the foveal periphery. Finally, there is also a bound on the information the human brain can visually process at any given time, but this is probably an ability that can be trained the most. In view of these natural limitations, which are bound to stay, we must devise ways to make the best use of the available retinal surface and cerebral potential, in light of the growing amount of visual information ready to be presented. These efforts have commonly been labelled focus+context techniques, where the resolution of the visual information presented is highest in the foveal center and then falls off towards the periphery in some smooth fashion, without performing any clipping of the viewed large object. Multi-resolution techniques, or even semantic zooms, can be employed to navigate across the resolutions in visual space, and a great number of techniques to control these have been described in the past, including various forms of lenses, warps, and distortions. On the other hand, there have also been a number of methods and metaphors to aid the user in the perceptional navigation of a dataset or object, such as stylized highlighting of features, cut-away views, and folding.

Interactive operability is the prime key to a successful user experience and his/her exploration and immersion in the data, and the GPU has provided an attractive platform to achieve these goals. Our work embraces this technology to provide a novel focus+context tool that unifies and extends a variety of existing methods in this area. Our techniques are primarily designed for volumetric objects, which have received the least amount of attention so far. Our framework provides a free-form volumetric lens function that can be feature-adaptive or user-configurable for a high-quality, anti-

*e-mail: {lujin|yezhao|mueller|ari}@cs.sunysb.edu

IEEE Visualization 2005
October 23-28, Minneapolis, MN, USA
0-7803-9462-3/05/$20.00 ©2005 IEEE.

aliased, and interactive display with smooth transitions from high- to low-resolution areas. It is somewhat related to the importance-driven visualization system, recently described by Viola et al. [22], but our method allows users not only to highlight and expose an object, but also to non-linearly magnify the object for closer inspection in its spatial and semantic context.

Our paper is structured as follows. We first present an overview of previous work on this subject, in Section 2, and then describe our volumetric lens, in Section 3 and GPU implementation in Section 4. Finally, we present results, in Section 5, and end with conclusions, in Section 6.

2 Related Works

Focus+Context Visualization Many techniques have been developed in this area. Zhou et al. [25] devised focus-region based volume rendering for volume feature enhancement. Volume data inside and outside the focus region are rendered in different styles, and the distance to the focal point is further included to control the optical properties of volume features in the context region [24]. Gaze-directed volume rendering [17] takes the observer's viewing focus into account to increase the rendering performance. The volume dataset is rendered at different resolutions, with the focal region represented at full resolution and the other parts at a lower resolution. Importance-driven volume rendering [22] is a view-dependent model for automatic focus+context volume visualization. The object importance is added as a new dimension to the traditional volume rendering pipeline in order to maximize the visual information. This technique removes or suppresses less important parts of a scene to reveal more important underlying information.

Cut-Away Views and Extensions Cut-away viewing, also known as volume cutting [20], is another way to display volumetric objects. Various cut-away techniques can be achieved automatically [7], and many improvements have been made. Instead of disposing cut-away volume parts, McGuffin et al. [18] use deformations for browsing volumetric data. Tory et al. [21] provide a framework, called ExoVis, for simultaneously viewing detail and context in volumetric data sets. It allows users to view multiple slices of a volume at arbitrary orientations, along with multiple subvolumes rendered in different styles. All slices and subvolumes are outside or surrounding a 3D overview of the dataset.

Lenses and Distortion Lenses in real world can be quite complicated [13]. However, simple lenses and magnifications are still very useful and have been thoroughly studied for text, image and information visualizations [16, 11, 12]. Bier et al. [1] introduced Toolglass and Magic Lenses as a see-through interface to modify the visual appearance of application objects, enhance data of interest or suppress distracting information. Viewpoint-dependent distortion of 3D data, see [3, 4] for example, highlights regions of interest by dedicating more space to them. On the other hand, relatively little work has been done on lenses in the domain of volume visualization. Cignoni et al. [5] provided the Magicsphere metaphor to visualize 3D data with a MultiRes filter. LaMar et al. [15] integrated a 3D magnification lens with a hardware-texture based volume renderer. Zooming is accomplished by modifying texture coordinates, and the 2D perspective correct textures technique is extended to 3D in order to obtain the correct texture coordinates for the lens border. Multiple segments on the border are needed to generate more natural circular lenses. Wei et al. [23] applied fisheye views to magnify particle track volume data using nonlinear magnification functions. Cohen and Brodlie [6] magnify volume data by generating a new volume using inverse distortion functions, however, this method is slow and is memory-intensive. Further research is clearly needed to design better lenses and find efficient implementations for volume data.

GPU-based Volume Rendering GPU-accelerated volume rendering can be based on textures [8] or ray casting [14]. Here we will not list all the papers on GPU-based Volume Rendering. Since our volume lenses are designed based on changes in ray direction or ray sampling rate, it is straightforward to implement, as well as extend, them using a ray casting approach.

3 Volumetric Lenses

In this section we describe several volumetric lenses which are based on geometric optics and conform to sampling theory.

3.1 Magnifier

The magnification lens, called magnifier in this paper, is based on the magnification model in optical physics. It provides users a method for close inspection of regions of interest in volumetric objects. Figure 2 illustrates the principle of a magnifier. The blue line segment represents a magnifier lens positioned on the image plane by the user. LC is the center point of the lens and F is the virtual focal point. When orthogonal incident rays hit the image plane, in the region of the magnifier, then the ray directions are modified and go through the focal point F. Therefore, a ray cone is formed between the lens and F. The objects within this cone are rendered in a larger area on the image plane than their original size, while the other objects retain their original size. Consequently, the objects in the region of interest are magnified.

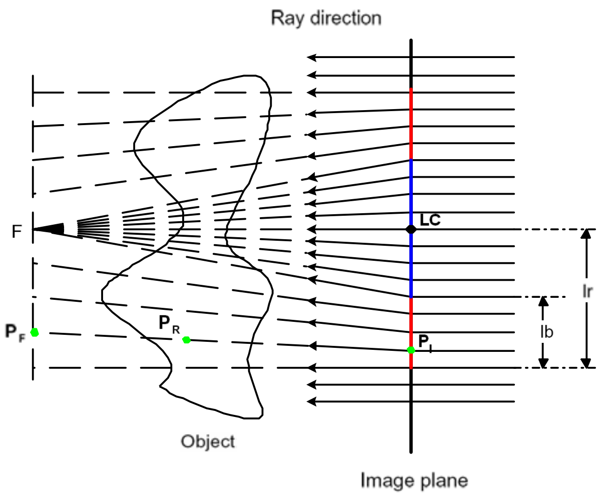


Figure 2: Magnifier illustration.

In the basic scenario described above, objects located between the orthogonal rays and the focused rays will not be visible on the image plane. This causes a loss of spatial context for the observed objects and has to be compensated for by special treatments. Our solution is to add a transition region close to the border of the ray cone where the directions of rays are gradually changed from the focused direction to the orthogonal direction. In Figure 2, the transition region is represented by the red line segments on the image plane with a width lb, lr is the radius of the lens, and the magnification region of the lens is shown as the blue line segment. For a ray starting from a point P_I in the transition region, the direction is computed according to the distance from P_I to LC as follows:

$$\frac{|P_F - F|}{lr} = \frac{|P_I - LC| - (lr - lb)}{lb}, \tag{1}$$

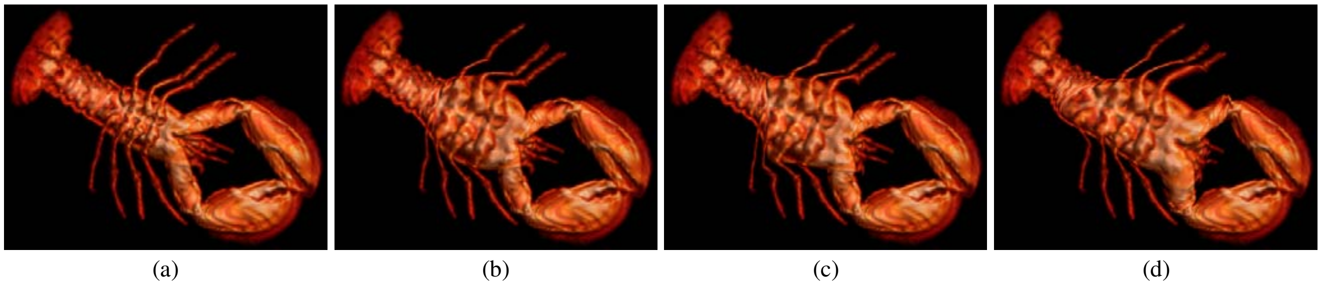


Figure 3: Magnifier volume renderings with (a) No lens, (b) Circular lens, (c) Square lens, (d) Arbitrary-shaped lens.

$$P_F = F + \frac{P_I - LC}{|P_I - LC|} \cdot |P_F - F|, \tag{2}$$

$$ray\_dir = P_F - P_I. \tag{3}$$

where P_F is the point at which this ray passes through the virtual lens focus plane, which is parallel to the image plane and includes the focal point F.

As a result of the transition region approach, while the objects inside the center region of the lens are magnified, the objects in the transition region are compressed. Therefore, continuous observation of the objects is achieved and no artificial data loss is introduced.

Based on this method, we are able to design magnifiers with any arbitrary shape. Results obtained by using magnifiers in volume rendering are shown in Figure 3. Figure 3a is the original volume rendering result with no magnifier and Figure 3b-d are the results obtained by using circular magnifier, square magnifier and arbitrary-shaped magnifier, respectively. Figure 4 shows the transition regions, magnification regions of three magnifiers, and the rendering effects on enlarged portions of Figure 3b-d.

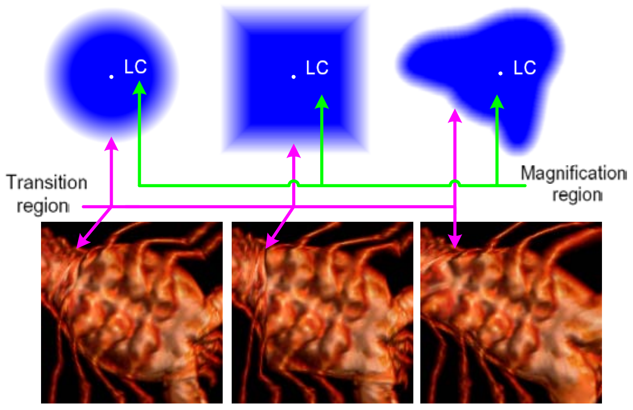


Figure 4: Transition region and its rendering effect.

The magnification factor can be changed by modifying the focal point position. Moving virtual focal point F towards the image plane achieves a higher magnification factor and vice versa. The GPU acceleration makes it possible for users to choose this interactively. At the same time, the users can also change the size of the magnifier, for example, the radius of a circular lens, and the size of the transition region to generate the desired results.

Our volumetric lenses are based on ray casting and it can be easily detected whether a ray pass the feature, therefore the magnifier can be utilized to enlarge only features of interest in the observed volumetric object. The magnification method is straightforwardly applied to the segmented volumetric datasets. The ray modification method does not interfere with the composition of the voxels with different properties because of their segmentation. Figure 5 shows the results of applying the magnifier to show the bone features of a segmented frog dataset.

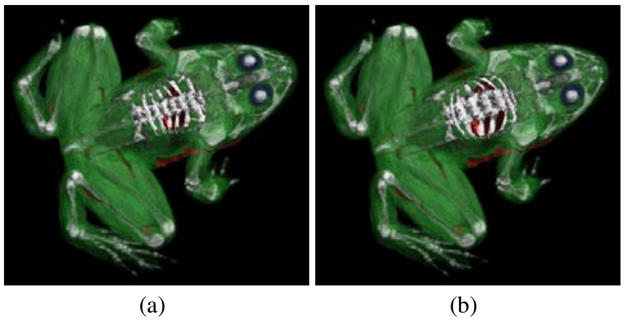


Figure 5: Magnifier volume renderings for the bone feature in a segmented frog dataset. (a) and (b) are renderings without and with magnification under circular lens.

Since in the transition region, the ray sampling rate is relatively low, aliasing could occur. Although this is not always noticeable in practice, anti-aliasing techniques can be applied to generate better results. A solution is to use volume texture mip-mapping to adaptively choose the appropriate resolution of the volume data for rendering. A lower resolution volume is chosen for regions sampled at a lower rate, in order to eliminate aliasing. One can determine the required mip-map level by calculating the magnification factor mf for point P_R,

$$mf = \frac{|P_R - P_{RI}|}{|F - LC|}\left(\frac{lb}{lr} - 1\right) + 1. \tag{4}$$

where P_{RI} is the orthogonal projection of P_R on the image plane. This factor will determine the mip-map level that needs to be used.

3.2 Feature-based lens

Feature-driven volume visualization provides users a highlighting and exposition of the portions of interest in volume objects. This facilitates an accurate and differentiated understanding of the important features. Besides the traditional fixed-shaped lens used to magnify segmented datasets, our free-form magnifier can be employed to also achieve a feature-sensitive and feature-centric object enlargement. The difference is that the shape of the magnifier is defined dynamically by the shape of the features (represented by the segmentation information) in the dataset, within an arbitrary view port. This is illustrated in Figure 6. Whether an incident ray changes direction depends on the distribution of the feature and the current view port. Thus the direction of each ray has to be determined dynamically. Transition regions are also used here to retain the space context of the features.

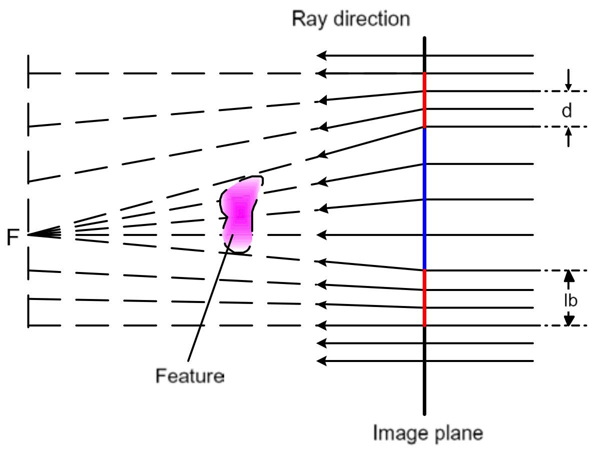
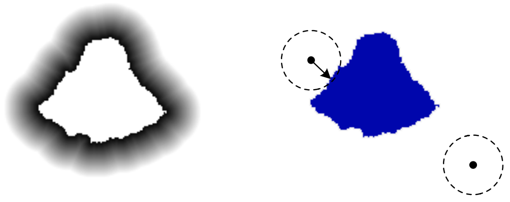


Figure 6: Feature-based lens illustration.

For each ray orthogonally incident upon the image plane, the new direction is computed as follows. Assuming all rays have changed directions to the focal point F,

- if a ray passes through the feature, then its new direction is pointing to F.
- if the ray does not pass through the feature but is inside the transition region on the image plane, the distance d (see Figure 6) from its entry point to the boundary of the feature-projected area is calculated. This distance is used to compute the new direction as in Equations1-3.
- otherwise, the ray continues along its original direction.

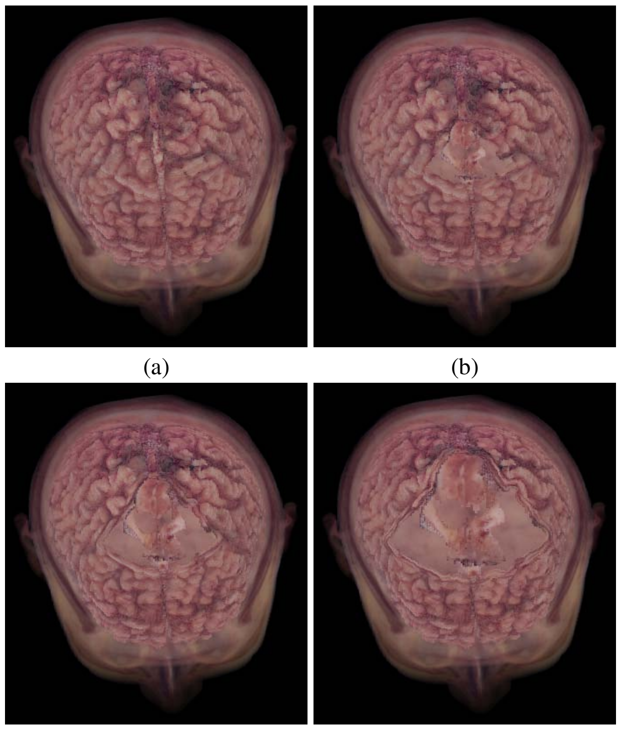


(a) (b)

Figure 7: Distance computation on the transition region of the feature-based lens. (a) Distance field on the transition region, (b) Searching circle for each pixel outside the feature-projected region is used for local computation.

On the image plane, the distance from a pixel to the boundary of the feature-projected area has to be calculated for some rays. This requires knowledge of the position of such an area on the image plane in each different view port. Therefore, a two pass computation has to be used, where the first pass defines the feature-projected region and the second pass computes the distance from a pixel to this region. Different distance computation methods can be used during the second pass. To facilitate the GPU acceleration for this algorithm, it has to be implemented based on local operations where each pixel only utilizes the knowledge of its neighborhood. Our implementation is to use a searching circle for each pixel with the transition region width lb as its maximal radius (see Figure 7 for an illustration). Inside this circle, we compute a neighbor that is a feature projected point and has the smallest distance to the pixel. This smallest distance is used as the distance value for this pixel. This method is implemented directly as a fragment program on GPU (see Section 4).

Our lens can be combined with any feature-based ray casting volume rendering method, for example, the two level volume rendering technique [9] for segmented volume data. Figure 8 shows some rendering results for a color volume dataset, in which a user selected feature is magnified and the other objects near that feature are compressed. Figure 8a shows the skin of the brain. Figure 8b shows an interior structure of the brain, without rendering other features which occlude this structure, while the magnified structures are shown in Figure 8c and 8d.



(a) (b)

(c) (d)

Figure 8: Feature-based lens volume renderings for a segmented human brain color volume dataset. (a) without specifying any feature of interest, (b) with a feature of interest, which is not magnified and appears too small to be seen clearly. From (c) to (d) the magnification factor increases.

3.3 Sampling-rate-based lens

We introduced two magnification lenses that modify the casted rays using geometric optics. They are implemented directly by changing ray directions from different areas of the image plane. The distribution of the areas can be user-defined or feature-based. In this section, we define a lens from another point of view. The rays casted towards the observed object may have varying densities in different portions of the object. This results in a varying sampling rate for the object. Therefore, this special lens is called *sampling-rate-based lens*. Various sampling functions could be adopted to define various volumetric lenses and to achieve different volume rendering results. we can use these lenses in conjunction with the mip-map volumes discussed in Section 3.1.

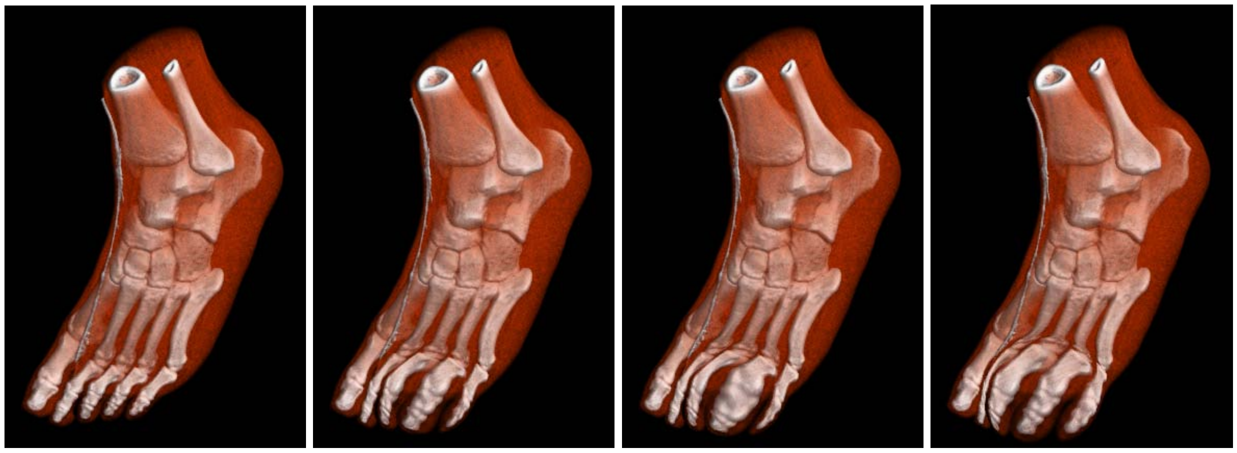

(a) (b) (c) (d)

Figure 10: Comparing volume renderings with (a) No lens, (b) Magnifier, and Sampling-rate-based lenses (c) Cubic sampling function (maximal sampling rate/normal sampling rate = 3), and (d) An arbitrary sampling function shown in Figure 11.

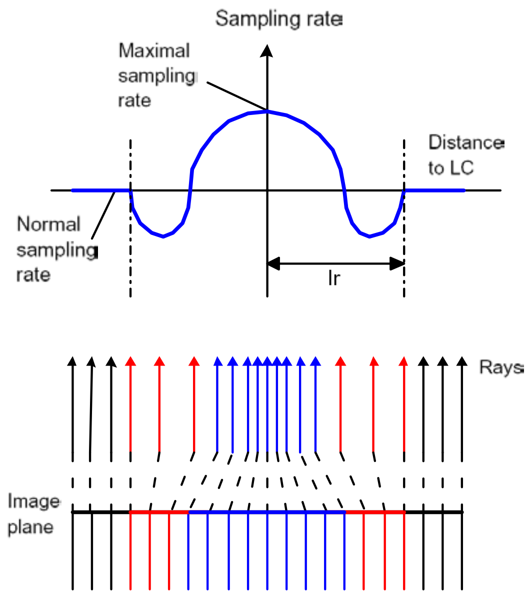


Figure 9: Sampling-rate-based lens illustration.

We illustrate the idea of this lens in 1D in Figure 9, where a sampling rate function is shown at the top and the corresponding rays are shown at the bottom. In the sampling rate function, lr is the lens radius, the vertical axis is the sampling rate and the horizontal axis represents the distance to the lens center. The sampling rate close to the lens center is the highest. It then decreases and becomes even smaller than the original normal sampling rate towards the boundary of the lens. At the bottom of Figure 9, we can see that the rays shot to the object are dense in the center region of the lens and become coarser towards the boundary. Note that the distribution of pixels on the image screen is uniform and that the original orthogonal rays are also distributed uniformly. To distribute the rays according to the sampling rate function, the start point of a ray is not from its original starting pixel but depends on its distance to the lens center and the sampling rate. Thus, we need to compute the correct start point for each ray. As usual, the transition region approach is applied to this lens. Here, the magnification region plus transition region must be exactly equal to the lens region, which means the distance from the cutting point (where sampling rate returns to normal) to the lens center must be equal to the radius of the lens, lr. Define sr as the sampling rate and sd as the sampling distance function. Here, sr is inversely proportional to the distance between sampling rays. We first precompute a coefficient C satisfying the integral equation:

$$\int_0^{lr} C \cdot sd(s)ds = lr, \tag{5}$$

$$sd(s) = \frac{1}{sr}. \tag{6}$$

Then for each ray j, the distance between its real start point and the lens center can be calculated using Equation 7, which is the discrete form of the distance integral.

$$distance(j) = \sum_{i=0}^{steps} C \cdot sd(i). \tag{7}$$

Figure 10 shows the results with the sampling-rate-based lenses, comparing it with the results obtained with no lens and with the magnifier. The toes of the foot are rendered with different magnification effects. The difference between Figure 10b and 10c is mainly caused by the different magnification factor distributions on the lenses. For the magnifier, the factors for points, which project into the magnification region and locate on the same plane parallel to the image plane, are the same. Therefore, objects with the same depth are magnified uniformly. However, for the lens with cubic sampling function, the factor is the highest on the lens center and decreases gradually towards the lens boundary. Objects with projections closer to the lens center are magnified with higher magnification factors. Along any ray, the factor remains the same for different depthes.

3.4 Angular lens

A common widely used lens is the fisheye lens [2], and our GPU accelerated general volumetric lens framework supports this type of lens as well. The fisheye lens is a specially designed lens which achieves wider viewing angles. The original fisheye lenses were photometric lenses designed to take photos of the entire sky. There are two main idealized fisheye projections, the hemispherical and

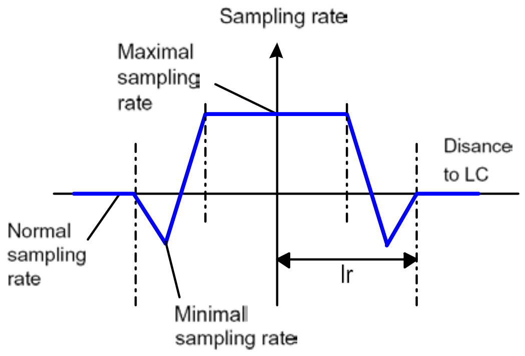


Figure 11: Another sampling rate function.

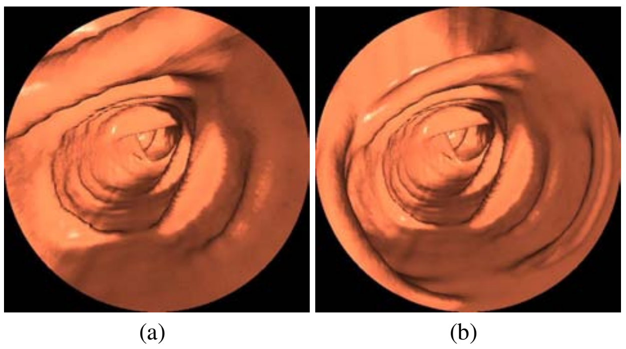


Figure 13: Virtual tour of the colon. (a) Perspective view with angle 120 degrees, (b) 180 degrees view with an angular fisheye lens.

the angular fisheye, which are common in computer graphics rendering [2]. The hemispherical fisheye is less used due to the distortion introduced. An angular fisheye projection can be used for angles up to 360 degrees and is defined such that the distance from the pixel P to the center of the image is proportional to the angle α of the viewing direction (see Figure 12a). The ray direction corresponding to any pixel on the image can be calculated by a special transform from pixel coordinates to 3D polar coordinates [2]. Figure 12b shows an image of a 180 degrees view on a bonsai with an angular lens.

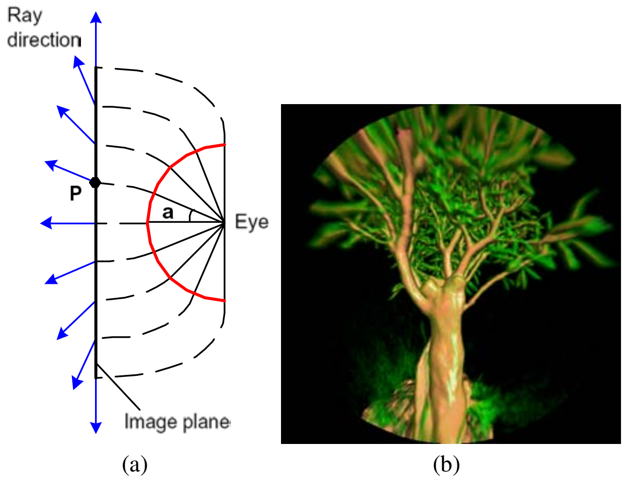


Figure 12: Angular lens. (a) Angular fisheye lens with 180 degrees illustration, (b) 180 degrees view of a bonsai with an angular fisheye lens.

Our framework is based on a ray casting volume rendering scheme. This allows us to walk into the interior of the object to see the augmented volume rendering results. By using an angular lens, larger view port angles can be achieved and more objects can be accommodated in the final image. This is helpful in many interior volume rendering scenarios. A good example is virtual colonoscopy [10]. When navigating inside the colon, more areas can be viewed to achieve a more efficient observation. Figure 13 shows the result of viewing a colon from a point on the centerline of the colon. Comparing this with a normal perspective view with 120 degrees, more information can be obtained when using a 180 degrees angular lens.

4 Hardware Acceleration

To achieve interactive focus+context volume rendering, we have implemented all of our volumetric lenses on contemporary graphic hardware. In GPU-accelerated ray casting volume rendering [14], front faces and back faces of the volume bounding box are drawn using OpenGL in two fragment passes to get the start and end points for all the rays. However, this approach can not be used for our volumetric lens. Because ray directions are not always orthogonal or perspective, we have to calculate the start and end points of each ray for the various lens algorithms described earlier. Hence, we implemented our own ray casting rendering algorithms with lens effects on the GPU. At first, we calculate the ray directions using the appropriate lens rules. Then, the intersection points of each ray with the bounding box of the volumetric object are computed. Finally, a ray traversal algorithm is implemented for a given step size, with the volume data (density, gradient or color) stored in 3D textures. All these algorithms are translated into Cg fragment programs. The current GPUs (e.g., NVIDIA GeForce 6800) have the required features, such as loop, early termination and branches, making it possible to implement our ray traversal method efficiently.

For our magnifier and angular lenses, we use four passes fragment programs as follows:

- *Pass 1: RayDirection* Calculate the ray direction for each fragment based on the view port and lens parameters. Also the information about whether a ray goes through the lens or hits the feature of interest, or the distance to the lens center can be obtained to achieve different rendering effects.
- *Pass 2: RayTfrontback* Compute the intersections of each ray with the volume bounding box, and store the distances from the front and back intersection points to the ray start point, denoted as *t_front* and *t_back*, which will be used along with the ray direction and view port parameters to define the intersection points in the next pass.
- *Pass 3: RayCasting* Cast the ray into the volume and composite the color based on the volume data and transfer function. Different traditional volume rendering modes can be easily added into this pass.
- *Pass 4: Rendering* Output the rendering results to the frame buffer.

For the feature-based lens, one more pass called *Pass 1+: RayLensBorder*, is added before *Pass 2*, to calculate the distance field for the lens transition region and change the ray directions based on the distance.

For sampling-rate-based lenses, ray directions are never changed, but the real ray start points need to be computed. We also use the four-pass fragment programs, but the first pass is changed to *Pass 1*: RayStartPoints*, which computes the ray start points used in later passes.

Table 1: GPU performance for different volume datasets.

Data	Data size	Volume lens method		Simple ray casting	
		Rendering speed (ms)	Frames/second	Rendering speed (ms)	Frames/second
lobster	$128 \times 128 \times 128$	70	14.2	61	16.4
engine	$256 \times 256 \times 110$	95	10.5	74	13.6
bonsai	$256 \times 256 \times 128$	110	9	95	10.5
foot	$154 \times 263 \times 222$	97	10.3	90	11.1
aneurism	$256 \times 256 \times 256$	186	5.4	158	6.3
frog	$502 \times 472 \times 138$	308	3.3	258	3.9

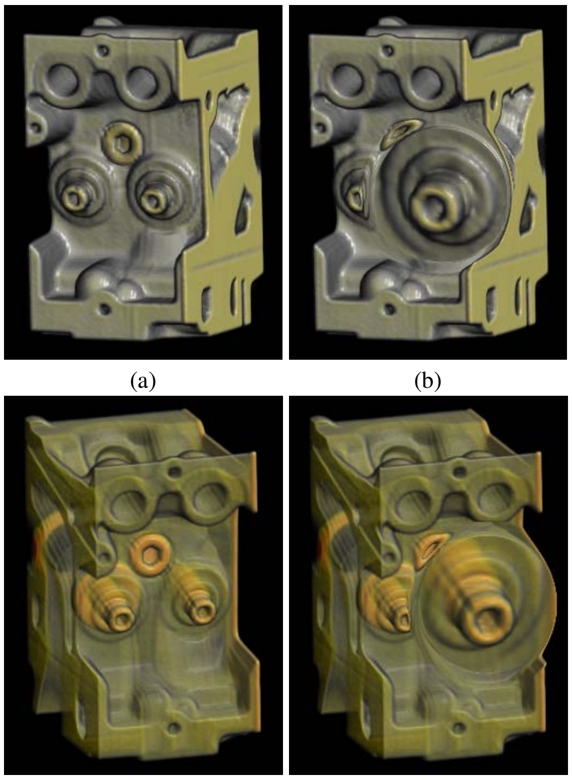


Figure 14: Magnification results. (a) and (b) are DVR results without and with magnifier, (c) and (d) are DVR with gradient magnitude modulation results without and with magnifier.

5 Results

We have implemented our methods on a Pentium Xeon 2.4GHz CPU with 2.5GB memory and an NVIDIA GeForce 6800 Ultra GPU with 256MB memory. In Table 1, we report the data size and the performance of our method with GPU-accelerated computation. For comparison, we also include the performance of a simple ray casting volume renderer (utilizing the front faces and back faces) with the same data sets on the same GPU. All the performances are tested with 512×512 images and with a 1.0 step size. Note that our method has not been optimized for the GPU, therefore, we compare it with the simple ray casting implementation, which is also unoptimized. Our volume lens methods only slightly increase the rendering time comparing to the general ray casting method. In the future, we will implement the standard optimization methods, such as empty space skipping to improve the performance. For example, the speed for aneurism data can be dramatically accelerated with space skipping.

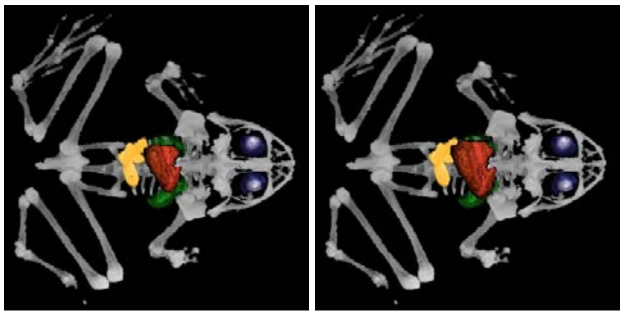


Figure 15: Feature-based lens results. (a)-(b) Feature frog heart is magnified, rendered with two level volume rendering method. The bone and eye retia are rendered with MIP, all other features are rendered with DVR, with different transfer function for each feature.

As a ray casting based augmentation for volume rendering, our volumetric lenses can be combined with many volume rendering modes, for example, direct volume rendering (DVR), MIP and DVR with no shading, DVR with gradient magnitude modulation, XRay and the two level volume rendering method for segmented data. We show results with several rendering methods in Figure 14, Figure 15 and Figure 16.

Our lenses can be used to interactively choose and magnify regions or features of interest to see small details more clearly while the context region remains. The size and shape of the lenses, and the magnification factor also can be changed interactively, which allows the user to adjust the lenses for desired results. Demo videos that show the interactive volume lens renderings can be obtained at http://www.cs.sunysb.edu/∼lujin/paper/vis05.

6 Conclusions

We have described a universal and general volumetric lens framework that has applications in many domains. It allows users to apply any well known lenses, such as a fisheye lens in the context of volumetric distortion, as well as design free-style and feature-adaptive lenses for arbitrary magnified focus+context viewing. For example, coupled with a GPU-based interactive segmentation algorithm it can be used to magnify the segmentation result at great detail and aid in its refinement. The support for free-style lenses, created with our lens design interface, can help illustrators to designed more helpful and informative visualizations of volumetric objects, emphasizing an arbitrary shaped region of interest without

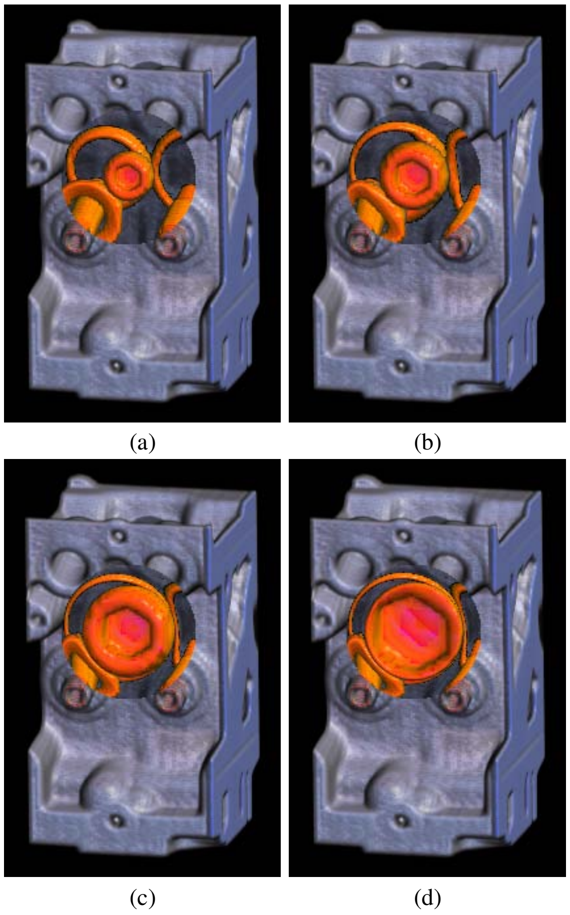


Figure 16: Feature magnification results with magnification factor increasing from (a) to (d).

losing the context of its surround. Finally, the GPU acceleration of our magic volume lens allows all of these to be done at interactive speeds, fostering both creative design and exploration.

In future work, we would like to extend this free-style zooming capabilities to multi-resolution data and to semantic zooms, where the data appearing under magnification comes from a different data source, or even texture synthesis. It may also proof helpful to users to provide an option for superimposing a lens-distorted lattice on top of the lens area, to aid in the assessment of the non-linear magnification effects.

Acknowledgements

This work has been supported by NSF grants CCR-0306438 and ACI-0093157. The datasets are courtesy of GE, National Library of Medicine, Lawrence Berkeley National Laboratory, Philips Research, Hamburg, Germany, Stony Brook University Hospital, and HuminTec Inc., Korea. Special thanks to Wei Hong and Feng Qiu for their contributions and valuable discussions in GPU based ray casting implementation, Satprem Pamudurthy for proof reading the paper, and the anonymous reviewers for their thoughtful comments.

References

[1] E. A. Bier, M. C. Stone, K. Pier, W. Buxton, and T. D. DeRose. Toolglass and magic lenses: The see-through interface. *Computer Graphics*, 27:73–80, 1993.

[2] P. Bourke. Computer generated angular fisheye projections. URL://astronomy.swin.edu.au/~pbourke/projection/fisheye/, 2001.

[3] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Distortion viewing techniques for 3-dimensional data. In *Proc. of IEEE Symposium on Information Visualization '96*, pages 46–53, 1996.

[4] M. S. T. Carpendale, D. J. Cowperthwaite, and F. D. Fracchia. Extending distortion viewing from 2D to 3D. *IEEE Computer Graphics and Applications: Special Issue on Information Visualization*, 17(4):42–51, 1997.

[5] P. Cignoni, C. Montani, , and R. Scopigno. Magicsphere: an insight tool for 3d data visualization. *Computer Graphics Forum (Proc. of EUROGRAPHICS '94)*, 13(3):317–328, 1994.

[6] M. Cohen and K. Brodlie. Focus and context for volume visualization. In *Proc. of Theory and Practice of Computer Graphics '04*, pages 32–39, 2004.

[7] J. Diepstraten, D. Weiskopf, and T. Ertl. Interactive cutaway illustrations. In *Proc. of EUROGRAPHICS '03*, pages 523–532, 2003.

[8] M. Hadwiger, C. Berger, and H. Hauser. High-quality two-level volume rendering of segmented data sets on consumer graphics hardware. In *Proc. of IEEE Visualization '03*, pages 40–47, 2003.

[9] H. Hauser, L. Mroz, G.-I. Bischi, and E. Gröller. Two-level volume rendering-fusing mip and dvr. In *Proc. of IEEE Visualization '00*, pages 211–218, 2000.

[10] L. Hong, S. Muraki, A. Kaufman, D. Bartz, and T. He. Virtual voyage: Interactive navigation in the human colon. In *Proc. of SIGGRAPH '97*, pages 27–34, 1997.

[11] T. Keahey and E. Robertson. Techniques for non-linear magnification transformations. In *Proc. of IEEE Symposium on Information Visualization '96*, pages 38–45, 1996.

[12] T. Keahey and E. Robertson. Nonlinear magnification fields. In *Proc. of IEEE Symposium on Information Visualization '97*, pages 41–49, 1997.

[13] C. Kolb, D. Mitchell, and P. Hanrahan. A realistic camera model for computer graphics. In *Proc. of the 22nd annual conference on Computer graphics and interactive techniques*, pages 317–324, 1995.

[14] J. Krüger and R. Westermann. Acceleration techniques for GPU-based volume rendering. In *Proc. of IEEE Visualization '03*, pages 287–292, 2003.

[15] E. LaMar, B. Hamann, and K. I. Joy. A magnification lens for interactive volume visualization. In *Proc. of the Ninth Pacific Conference on Computer Graphics and Applications*, pages 223–232, 2001.

[16] Y. K. Leung and M. D. Apperley. A review and taxonomy of distortion-oriented presentation techniques. *ACM Trans. on Computer-Human Interaction*, 1(2):126–160, 1994.

[17] M. Levoy and R. Whitaker. Gaze-directed volume rendering. *Computer Graphics (Proc. of Symposium on Interactive 3D Graphics '90)*, 24(2):217–223, 1990.

[18] M. J. McGuffin, L. Tancau, and R. Balakrishnan. Using deformations for browsing volumetric data. In *Proc. of IEEE Visualization '03*, pages 401–408, 2003.

[19] R. Perlman. *Light and color*. Golden Press, 1971.

[20] B. Pflesser, U. Tiede, and K. H. Höhne. Towards realistic visualization for surgery rehearsal. In *Proc. of Computer Vision, Virtual Reality and Robotics in Medicine*, pages 487–491, 1995.

[21] M. Tory and C. Swindells. Comparing exovis, orientation icon, and in-place 3d visualization techniques. In *Proc. of Graphics Interface '03*, pages 57–64, 2003.

[22] I. Viola, A. Kanitsar, and M. E. Gröller. Importance-driven volume rendering. In *Proc. of IEEE Visualization '04*, pages 139–145, 2004.

[23] X. Wei, A. E. Kaufman, and T. J. Hallman. Case study: visualization of particle track data. In *Proc. of IEEE Visualization '01*, pages 465–468, 2001.

[24] J. Zhou, A. Döring, and K. D. Tönnies. Distance based enhancement for focal region based volume rendering. In *Proc. of Bildverarbeitung für die Medizin '04*, pages 199–203, 2004.

[25] J. Zhou, M. Hinz, and K. D. Tönnies. Focal region-guided feature-based volume rendering. In *Proc. of 1st International Symposium on 3D Data Processing Visualization and Transmission*, pages 87–90, 2002.