Sphere Light Field Rendering

Zigang Wang[1] and Zhengrong Liang[1,2]

Departments of Radiology[1] and Computer Science[2], State University of New York, Stony Brook, NY 11794, USA

ABSTRACT

Light field algorithm is one of the most famous image-based rendering techniques. In this paper, an improved light field algorithm -- sphere light field algorithm -- is proposed. This new algorithm replaces the parallel planes used in traditional light field algorithms by a triangularly parameterized sphere surface. Comparing to the traditional light field algorithms, this new algorithm achieves a more uniform distribution of light slabs in the three dimensional space. This improves the quality of the rendered images. Using the triangular parameterization and subdivision of the whole light field, less calculation and less memory are needed, resulting in improved real-time rendering.

Keywords: Image-based rendering, Light field, Volume rendering.

1. Introduction

Image-based rendering (IBR) techniques are a new approach to rendering a three dimensional (3D) virtual environment. Unlike the conventional model-based 3D rendering approaches, IBR techniques recover the environment from the dataset of the photographs and images. Because IBR techniques do not use both the complex 3D object model and the global illumination model to create the final image, a large amount of visibility calculation will not be needed. Furthermore, the IBR techniques are independent from the complication of the virtual environment. Therefore, the rendering time becomes a constant, eliminating the environment effect.

In 1993, Chen and William [4] proposed an image-based rendering technique for image interpolation and synthesis between the images and the pixels on the images from different view. One of the earliest systems using the image-based rendering technique is QuickTime VR [5]. In QuickTime VR, a 360-degree cylindrical panoramic image is used to create the virtual environment. It allows users to look around in the virtual environment. In 1995, Levoy and Hanraham [10] and Gortler, *et al*. [6] presented a new comprehensive description of the whole environment -- light field or lumigraph. Using the parameterization of the parallel planes, their light field algorithm uses light slabs to reconstruct the whole 4D plenoptic function. Because this algorithm is simple and comprehensive, it is suitable for the rendering of the whole environment, which is viewed from all view and all position out of the convex boundary. To improve the light field or lumigraph, Sloan and Hansen [11] presented a new parallel technique to reconstruct the lumigraph or light field in a parallelly distributed share memory computer. Isaksena, *et al*. [8] proposed a new dynamical reparameterization algorithm for the light field, which allows interactively rendering of the moderately sampled light field and autostereoscopic viewing. Further improvement is currently an attractive research topic.

Because the 3D volume data have a simple geometry structure, and it needs to be observed from different views and angles, so the light field algorithm has an outstanding advantage in the volume environment rendering. In the light field rendering for 3D volume, a uniform distribution of the light slabs is crucial. Non-uniform distribution will result in blurring effect in some section of the final rendering images. Traditional light field or lumigraph uses pairs of planes to implement the parameterization of the plenoptic function, it is difficult to get a uniform distribution of the light slabs in the light field. To solve this problem, we proposed a new image-based rendering technique -- sphere light field. This algorithm replaces the parallel planes with one sphere surface. After parameterization of the sphere surface, the light

[1] For correspondence: zgwang@clio.rad.sunysb.edu; telephone: (631) 444-2508. This work was partly supported by a NIH grant # CA82402 of National Cancer Institute and grant #HL51466 of national Heart, Lung, and Blood Institute.

slabs distribute uniformly in the given space. By introducing the bi-triangle interpolation, this algorithm can achieve the same quality of the rendered images as the traditional methods at a much less cost of computing effort.

The content of this paper are organized as follows. In section2, we propose the parameterization of the sphere surface and construction of the whole sphere light field. This is the core idea of our new algorithm. In section 3, the rendering of the whole environment from the sphere light field is presented. Using the triangle interpolation method, this algorithm remains the high quality of the rendered images at a less cost of computation. To implement the algorithm on currently available PC platforms with limited memory, a sub-field strategy is proposed in section 4. By subdividing the whole field into several small sub-fields, it only loads the whole dataset into the memory at one rendering time.

2. Plenoptic Sphere

In the light field, a plenoptic function is reconstructed by a set of light slabs [10] across each pair of planes. The light slab is represented as the beam of light entering one quadrilateral in one plane and exiting from another quadrilateral in another plane (Figure. 1). Using several pairs of planes, we can achieve a whole light field dataset. The main advantage of this presentation is the efficiency of geometric calculation [10], either the mapping points in the first plane (UV plane) to the points in the latter plane (ST plane) or the inversing mapping is projective mapping. Usually the light slabs constructed by the planes cannot distribute uniformly in the given 3D space. It is difficult to adjust the position and the angle of two planes to get uniform distribution.

Our new algorithm overcomes the difficulty. In our method, the planes are replaced by one sphere surface. Thus, the light is redefined as that the beam of light enters at one point on the sphere surface and exits from another point in the same sphere surface (Figure. 2). In the following sections, this light slab will be called as “sphere light slab”.

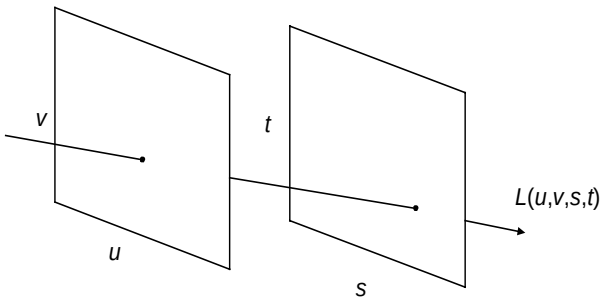


Figure 1: The light slab representation in traditional light field.

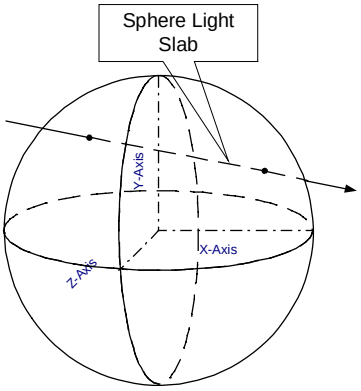


Figure 2: The sphere light slab representation.

Comparing with the traditional approaches of using parallel planes, our method of using sphere surface to construct the light field has some advantages:

It has a more uniform light slab distribution: When applying the traditional light field algorithm for real-time rendering, the view may be at any position and by any angle. If the distribution of the light slabs is not even or uniform in the space, there will be unevenly blurring at a specific view position than at other positions. This will cause the whole rendering process less consistent. Using a sphere surface, the distribution of all light slabs will be more even or uniform in the specific 2D/3D space, thus results in a more consistent rendering.

It is simple and needs less space than the planes: The nature of the sphere determines that the distribution of the light slabs only depends on the radius of the sphere. Comparing to the planes, the determination of the sphere surface is more simple and easier. At the same time, it needs less light slabs to achieve the same result as the traditional methods of using parallel planes.

It does not add more geometric calculation: Just same as that for parallel planes, the calculation of a sphere is fixed. It does not increase the calculation effort as the planes do. Using the bi-triangle interpolation, our new approach obtains the same rendered images at a less calculation cost.

2.1. Parameterization of the Sphere Surface

As defined above, a sphere light slab enters into one point on the sphere surface and exits from another point on the same sphere surface. The whole light field is composed of all available light slabs. So parameterization of the sphere surface is necessary. After parameterization, several sample points will be selected from the sphere surface, our algorithm only considers about those light slabs across the sample points.

2.1.1. Terms and Background

First, the whole sphere surface is divided into eight sections, each section is called "octsphere". Because all octspheres are similar, without losing the generality we will use section octsphere V($x > 0$, $y > 0$, $z > 0$) as an example to describe the parameterization.

In octsphere V, there exist three points A, B, and C, which are the intersections of the sphere surface with the three coordinate axes (see Figure 3). All these three points construct a 3D triangle ΔABC. The plane on which the triangle ΔABC lies on is called triangle plane. Given a point P on the octsphere, there exists a point P' on the ΔABC triangle plane, this point is called the project point of the point P. Obviously, for each point on the octsphere, there must exists a project point on the triangle plane; for each point p on the triangle plane, it also must exists a point on the octsphere whose project point is point p. So this mapping is one-to-one.

Furthermore, given two points P_1 and P_2 on the sphere surface, assume the points P_1' and P_2' are the mapping points of these two points. The corresponding mapping of the large arc (the circle whose center is exactly the center of the sphere) P_1P_2 on the triangle plane is the line section $\overline{P_1'P_2'}$ (see Figure 3).

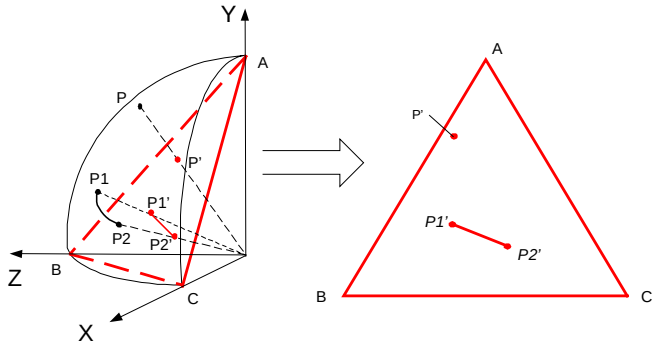


Figure 3: The mapping between the octsphere and triangle planes.

2.1.2. Parameterization of the Octsphere

For triangle ΔABC, let point B be the original point, and line BC and BA be two axis of the coordinates, see Figure 3. Then we construct a new 2D triangle coordinate. All points in the triangle ΔABC can be represented as

$$\{(u, v) \mid u, v \in [0,1] \quad \text{and} \quad 0 <= u + v <= 1\}$$

As described above, the mapping between the triangle ΔABC and Octsphere is one-to-one. Thus any point P on the octsphere can be parameterized as: the parameter of point P is (u,v) if and only if the parameter of P' on the triangle coordinate is (u,v) (see Figure 4).

Given any point $p(x,y,z)$ on the octsphere, the new parameters of the $p(u,v)$ are:

$$u = x / (x+y+z)$$
$$v = y / (x+y+z) \qquad (1)$$

If the $\overline{BC}$ and $\overline{BA}$ are divided into N similar sections, the whole region of triangle (u >=0 and v>=0 and $u + v$ <= 1) can be divided into (N+1)*N / 2 small triangles (see Figure 4). All intersection points are the sample points, which will be used to construct the whole light field.

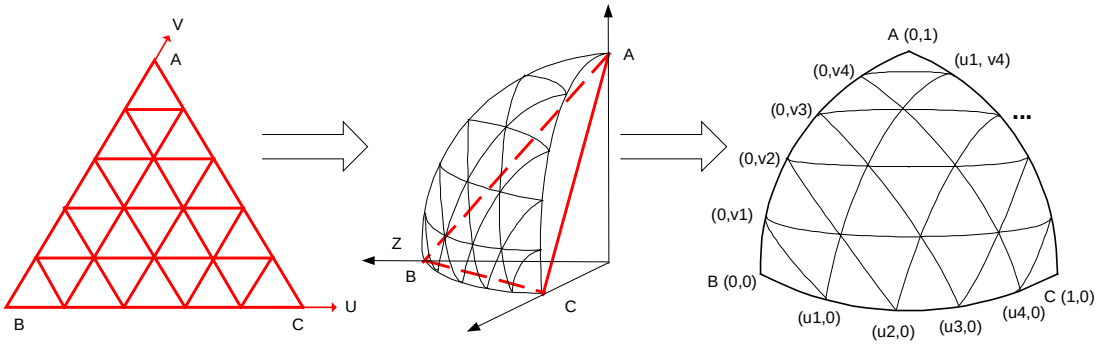


Figure 4: The parameterization of the sphere surface.

2.2. Sphere Light Fields

The whole sphere light field is composed of all sphere light slabs across the sample points on the sphere surface. This can be expressed as following:

$$\mathrm{LSF}(u, v, s, t) = \{\, l \mid l \text{ enters point } (u,v) \text{ and exits from point } (s,t) \,\}$$
$$u, s \in [0,1],\ v \in [0, 1-u],\ t \in [0, 1-s]$$

For each light slab, it is assigned a color, which represents the final rendering result on this light slab. After collecting all colors of the light slabs, all colors can be sorted by an image array. Thus all light slabs from the same sample points are grouped together. As for the method to achieve the colors of the light slabs, there exist several ways. In the virtual environment, traditional rendering algorithms can calculate the final color results of the light slabs. In real environment, the colors can be recorded by a digital camera or other photograph instruments. In this paper, we focus on the virtual environment, especially on the 3D volume environment. In the 3D volume environment, each light slab is viewed as a ray cast from the view. The traditional ray-based volume-rendering algorithms [2, 9] can generate all the colors. There also exist many speeded ray-tracing algorithms [1, 7], which use the coherence of adjacent ray to accelerate the speed of the rendering.

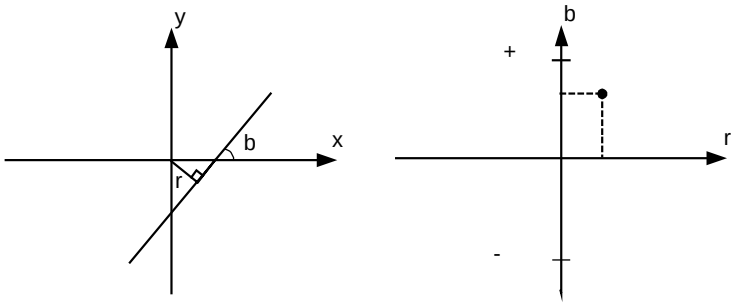


Figure 5: The definition of the line space. On the left, the line is defined in Cartesian space; on the right, this line is defined as a point in the line space.

In the line space [10], it is easy to evaluate the distribution uniformity of the light slabs. In the line space, each line is represented as (r, b): r is the distance from the original point to the line, b is the angle of the line, as showed in Figure 5. In the line space, a good distribution of the light slabs should be: (a) all regions of the line space should have the same density; (b) the points should cover the whole range of b $[0, 2\pi]$; and (c) the width of the points set in r direction should be as wide as possible.

In Figure 6, the light slabs, constructed from 2D lines (traditional method) and a circle and its corresponding display in the line space (our approach), are compared. It is just an example to compare the distributions of the light slabs between two methods. As showed in this figure, the distribution of the light slabs constructed from the lines is less uniform than those constructed from the circle. Furthermore, for the light slabs constructed from the circle, the width of the points set in axis r is approximately equal to the diameter of the sphere (see Figure 6b). By modifying the radius of the sphere, the distribution of the light slabs can be adjusted.

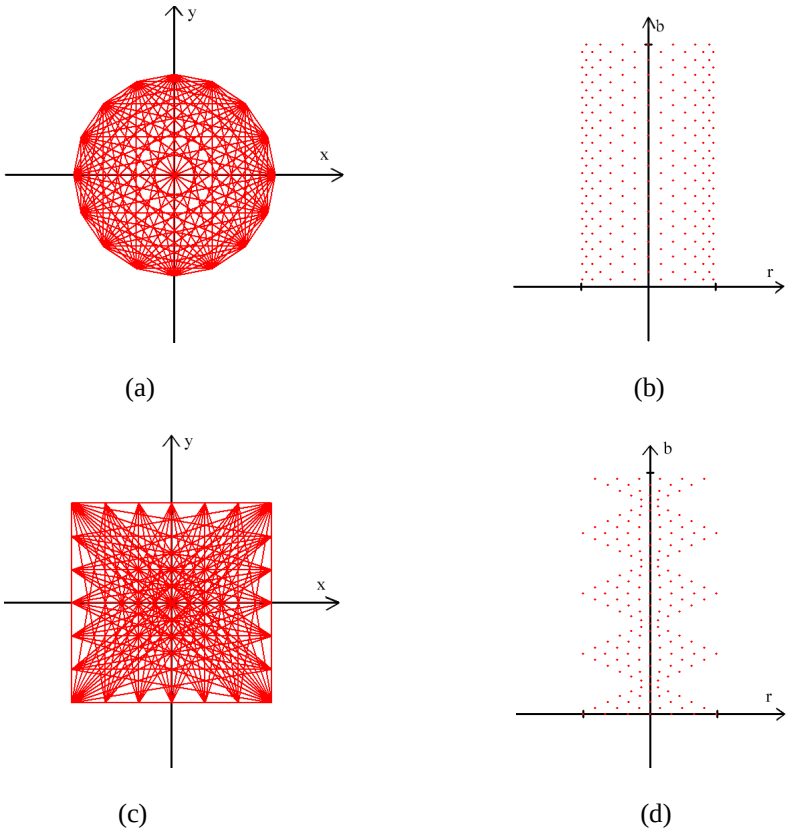


Figure 6: Comparison of the coverage of the light slabs in the line space. (a) shows the light rays constructed by the sphere. (b) shows the corresponding display of the lines in the line space. (c) and (d) show the light ray constructed by a pair of planes and its corresponding display in the line space.

3. Rendering of the Project Images

When the view position and view orient are determined, then for each pixel on the project image there exists a ray cast from that view. If this ray intersects with the sphere surface, the two intersect points are then given. If these two points are superposed with two sample points on the sphere, the light slab across these two sample points is the ray from that pixel. Then the color of that pixel is assigned by the color of this light slab.

In most circumstance, the intersecting points are not superposed upon any sample point on the surface. Then interpolation is necessary. Usually an intersecting point locates within a triangle whose three corner points are the sample points (see Figure 7). Then a triangle interpolation method is used in reconstructing the whole field. Given a

point P on the sphere surface, it locates within the triangle of three points: P_1, P_2, P_3 (as showed in Figure 7b). The P can be reconstructed by the following equation:

$$P = P_1 * (1 - \Delta u - \Delta v) + P_2 * \Delta u + P_3 * \Delta v$$

where

$$\Delta u = (u - u0) / (u_1 - u_0)$$
$$\Delta v = (v - v_0) / (v_1 - v_0) \qquad (2)$$

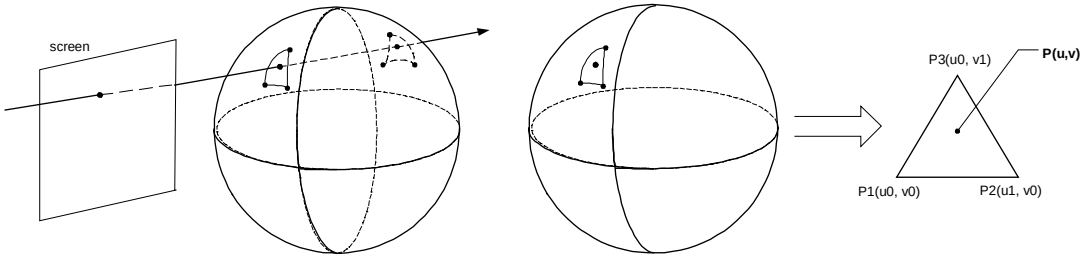
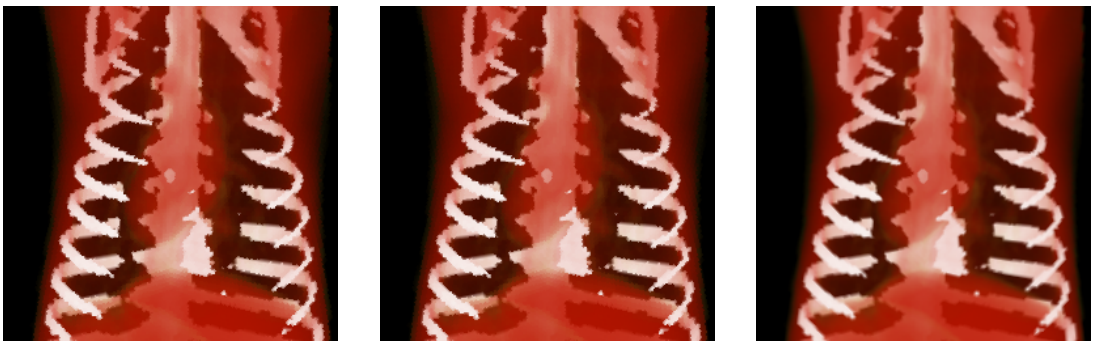


Figure 7a: A ray from a view intersecting the sphere. **Figure 7b**: Reconstruction of the intersecting point with the triangle interpolation.

Comparing to the bi-linear interpolation used in the plane light field, the triangle interpolation needs less calculation. It also can save more time in the real-time rendering process. To get the color of this ray, there exist several alternative methods: (1) it assignes the color of the closest light slab to the pixel; (2) it interpolates solely from *uv* or *st*; (3) it uses bi-triangle interpolation for *uvst*. In Figure 8, several different interpolation methods are compared. Apparently, the result of the bi-triangle interpolation is the best.



(a) (b) (c)

Figure 8: Comparison of different interpolation results. (a) shows the result without any interpolation; (b) is the result with the interpolation only in *uv* parameter; and (c) is the resulted image with the bi-triangle interpolation.

4. Subdivision of the Sphere Light Field

Because sphere light field contains the information of the dataset from all views, its size can be very large. In general, the size of the dataset is more than 1G bytes. It is expensive to load so large data set into the memory at one time. Actually, we need not load all data at one time. Once given a view position and orient, it only needs a small section of the whole data set to reconstruct the final image.

As described in section 2, the whole sphere field is divided into eight octspheres, thus all sample points on the sphere surface are classified into eight classes P_i:

$$P_i = \{ p \mid p \text{ is sample point and } p \text{ on the section } i \} \qquad i \in [1, 8].$$

Then the whole light field L can be divided into 64 small section $L_{i,j}$, it is called subfield:

$$L_{i,j} = \{ \text{ light slabs } l \mid l \text{ enters a sample point in } P_i \text{ and exits from a sample point in } P_j \} \quad i, j \in [1, 8].$$

When the view position and view orient are concerned, only the sub-fields whose light slabs will be used are loaded into the memory. Usually, the total amount of sub-fields loaded at one time is less than 16. That is, no more than 1/4 whole sphere light field will be loaded at one time.

If the size of the sub-fields is still large, the subdivision of the sub-fields is necessary. One section can be subdivided into four or more small subsections by three arcs (see Figure 9). Thus, the whole sphere surface will be divided into 52 sub-sections after the second subdivision. The more subdivision made, the less memory needed during the rendering.

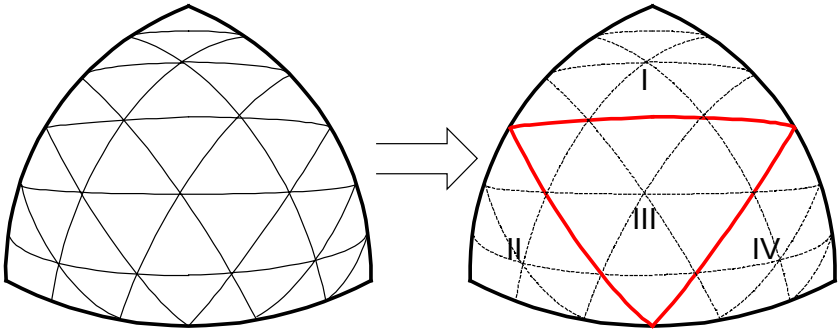
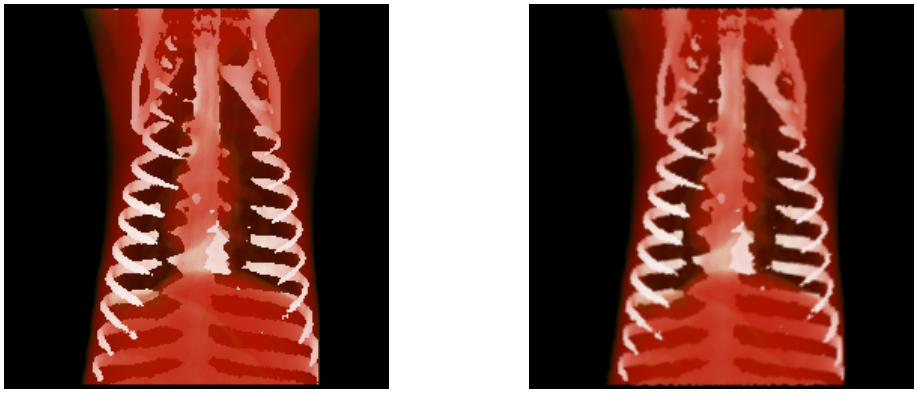


Figure 9: The subdivision of the octsphere.

Although sub-fields can lower down the cost of memory, it cannot decrease the size of the whole field data. Data compression techniques can be used to solve this later problem. There are many compression algorithms available in public domain, such as JPEG/MPEG. All these algorithms can be used in our sphere light field approach.

5. Experiments and Results

The entire sphere light field algorithm presented above was implemented in Visual C++ on a SGI visual 320 PC workstation. During the rendering of the sub-fields, two process are created: one is the normal rendering process and its function is to render the final image; and the other is a data loading process and this process can forecast the position and the orientation of the view in next frame, so it loads the blocks of sub-fields in advance and free the unused sub-field blocks. Since this process is independent from the other process, it can run with another process simultaneously, leading to speed improvement in the real-time rendering.



(a) (b)

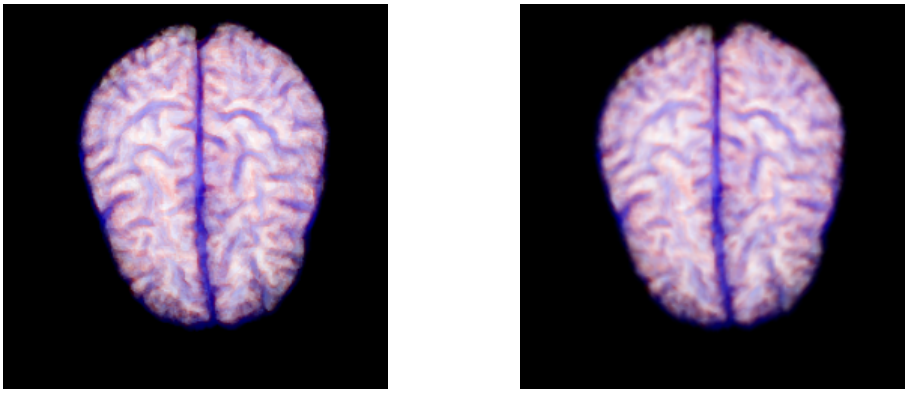
Figure 10: Rendered project images from the otter data. (a) shows the final image generated by the ray tracing volume rendering. (b) is the final image generated by our sphere light field.

The first example showing our sphere light field approach is a volume data of an otter body. The data is acquired by a CT (computed tomography) scanner from the otter. The volume data has an array size of 128x128x197 voxels. Figures 10a and 10b display the rendered images from the original ray-tracing volume rendering algorithm and our sphere light field, respectively. Table 1 lists the rendering times for one image using different algorithms. From the list, we can see that the calculation time of our sphere light field is approximately the same of the plane-based light field technique. Both of them are much faster than the original ray-tracing volume rendering algorithm.

	Data Size	Rendering time using volume rendering	Rendering time using sphere light field	Rendering time using plane light rendering
Otter Body	128*128*139	3.576 s	0.116 s	0.137 s
Segmented Brain	256*256*64	8.652 s	0.117 s	0.144 s

Table 1: Comparison of rendering times using different algorithms.

Our second example is a 3D volume data of the brain. This data is a post-segment dataset. Its size is 128x128x64 voxels. Our sphere light field result is illustrated in Figure 11(b). Corresponding volume rendering image is showed in Figure 11(a). The resolution of all the images in Figure 10 and 11 is 256x256, down sized to 128x128. The rendering times are listed in table 1. Comparing to the image generated by the volume rendering, the image generated by the light field or sphere light field has very little blurring effect. This phenomenon will disappear when the sample density reaches an upper-limit. The details about the samples are described by Chai [3], and are omitted here.



(a) (b)

Figure 11: Rendered project images from the brain. (a) shows the final image generated by the ray tracing volume rendering. (b) shows the final image generated by our sphere light field.

6.Conclusion and Future Work

In this paper, we introduced a new algorithm for image-based rendering of a 3D volume environment. This algorithm is based on the light field algorithm. This algorithm uses the sphere surface and triangle parameterization to reconstruct the light field, so it can overcome several shortcomings of the plane-based light field. Comparing to the plane-based light filed algorithm, the distribution of light slabs is more uniform by our sphere surface light field. Thus the quality of rendered images can be improved. The project image from arbitrary angle can be created in real-time by combining and re-sampling the whole sphere light field. Applying sub-fields can reduce the memory demand.

Although our algorithm is powerful in rendering the 3D volume environment, it has some limitations. First, the size of a dataset can be very large, for example more than 1G bytes. With the increase of the sample resolution, the size of the dataset will increase rapidly. Although the sub-fields and the compression techniques can be utilized, the problem may remain. Second, our algorithm only is suitable for the rendering of the environment with fixed light. If the light position and attribute changes dynamically, our algorithm may hardly handle the dynamical situation. Our future research will focus on the solutions of these limitations.

References

1. S. J. Adelson and C. D. Hansen, "Fast Stereoscopic Image with Rau-Traced Volume Rendering", *Proceedings Symposium on Volume Visualization*, pp. 3-9, October 1994
2. James Arvo and David Kirk, "Fast Ray Tracing by Ray Classification", *Proceedings of the 14th Annual Conference on Computer Graphics*, pp. 55-64, July 27 - 31, 1987
3. Jin-Xiang Chai, *et al.*, "Plenoptic Sampling", Computer Graphics, *Proceedings of Annual Conference Series (Siggraph'2000)*, pp. 307-318, 2000
4. Shenchang E. Chen and Lance Williams, "View Interpolation for the Image Synthesis", Computer Graphics, *Proceedings Annual Conference Series (Siggraph'1993)*, pp. 279-288, 1993
5. Shenchang E. Chen, "QuickTime VR – An image-based approach to virtual environment navigation", Computer Graphics, *Proceedings Annual Conference Series (Siggraph'1995)*, pp. 29-38, 1995
6. Steven J. Gortler, Redek Grzeszczuk, *et al.*, "The Lumigraph", Computer Graphics, *Proceedings Annual Conference Series (Siggraph'1996)*, pp. 43-52, 1996
7. Taosong He and Arie Kaufman, "Fast Stereo Volume Rendering", *Proceedings of the Conference on Visualization '1996*, pp. 49-56, 1996
8. Aaron Isaksen, *et al.*, "Dynamically Reparameterized Light Fields", Computer Graphics, *Proceedings of Annual Conference Series (Siggraph'2000)*, pp. 297-306, 2000
9. Marc Levoy, "Efficient Ray Tracing of Volume Data", *ACM Transactions on Graphics*, vol. 9, no. 3, pp. 245-261, July 1990
10. Marc Levoy and Pat Hanraham, "Light Field Rendering", Computer Graphics, *Proceedings Annual Conference Series (Siggraph'1996)*, pp. 31-42, 1996
11. Peter P. Sloan and Charles Hansen, "Parallel Lumigraph Reconstruction", Computer Graphics, *Proceedings Annual Conference Series (Siggraph'1999)*, pp. 7-14, 1999