# Fast Stereoscopic Images with Ray-Traced Volume Rendering

Stephen J. Adelson* Charles D. Hansen†
Los Alamos National Laboratory

## ABSTRACT

One of the drawbacks of standard volume rendering techniques is that it is often difficult to comprehend the three-dimensional structure of the volume from a single frame; this is especially true in cases where there is no solid surface. Generally, several frames must be generated and viewed sequentially, using motion parallax to relay depth. Another option is to generate a single stereoscopic pair, resulting in clear and unambiguous depth information in both static and moving images.

Methods have been developed which take advantage of the coherence between the two halves of a stereo pair for polygon rendering and ray-tracing, generating the second half of the pair in significantly less time than that required to completely render a single image. This paper reports the results of implementing these techniques with parallel ray-traced volume rendering. In tests with different data types, the time savings is in the range of 70 - 80%.

## 1. INTRODUCTION

Ray-tracing is a common method for generating realistic images of complex surfaces. A similar technique, sometimes referred to as ray-casting, is also frequently used with volumetric data. Rays are cast through the volume from each pixel, accumulating color and opacity as they travel, and returning an overall color for that position. Like all volume rendering techniques, however, a single image can be difficult to interpret because of image artifacts which hinder structure determination. The problem is even more complicated when there is no solid surface and color is used to represent some characteristic of the data rather than a lighting function. Real-world depth cues such as luminosity, perspective, shading, and occlusion are, at best, difficult to use in many volumetric images, particularly those with no isosurface [2].

The general solution to this problem is to generate multiple images of the same data from displaced view points and view them sequentially. Using motion parallax in this way is an effective depth cue, but it has two major shortcomings. First, multiple images of complex data involve significant allocation of both time and computing resources. Second, depth information is only available while the image is in motion. Ideally, a volume could be paused in its movement and examined. If the progression of views is halted, the motion depth cue is lost.

Another solution to the same problem is to generate a single stereoscopic pair. Stereoscopy gives unambiguous depth information when it is the only available depth cue, and depth discrimination is enhanced when other cues are also included [3, 7, 11, 17, 18, 19, 21, 24]. Only two images need to be rendered, the two halves of the stereo pair, and no motion is necessary; static images can be examined with no loss of perceived structure. In this paper, it is shown that both halves of a stereoscopic pair of ray-traced volumetric images need not be fully rendered. Exploiting the coherence between the two views, the second half of the stereo pair can be generated in a fraction of the time of the first half.

* MS M986, Los Alamos, NM 87545 (adelson@lanl.gov)
† MS B287, Los Alamos, NM 87545 (hansen@acl.lanl.gov)

0-8186-7067-3/95 $4.00 © 1995 IEEE

## 2. BACKGROUND AND PREVIOUS WORK

### 2.1 Stereoscopic Reprojection

Most volume renderers use parallel projections for generating images. In this section, stereoscopic projection is derived for parallel projections, although it is a fairly straightforward matter to derive for perspective projection [1, 10].

The most efficient viewing geometry for parallel stereoscopic imaging is illustrated below in Figure 1. There are two view points, separated by a distance **e**: the left center of projection (LCoP) for the left-eye image and the right center of projection (RCoP) for the right-eye image. Instead of placing the centers of projection symmetrically about the origin (as they would be in traditional rendering geometries), one is positioned at the origin and the other at the point ($\mathbf{e}\cos(\Phi/2)$, 0, $\mathbf{e}\sin(\Phi/2)$). This is accomplished by changing the viewing transformation matrix so that the center of the viewing coordinate axes is at ($\mathbf{e}/2\cos(\Phi/2)$, 0, $\mathbf{e}/2\sin(\Phi/2)$) and the w-axis of the (u, v, w) viewing coordinate system is ($-\sin(\Phi/2)$, 0, $\cos(\Phi/2)$). This makes the center of rotation $\mathbf{P} = (0, 0, \mathbf{R})$, where $\mathbf{R} = \mathbf{e}/[2\sin(\Phi/2)]$. No additional calculations are necessary to view-transform data when this change is made in the viewing transformation matrix.

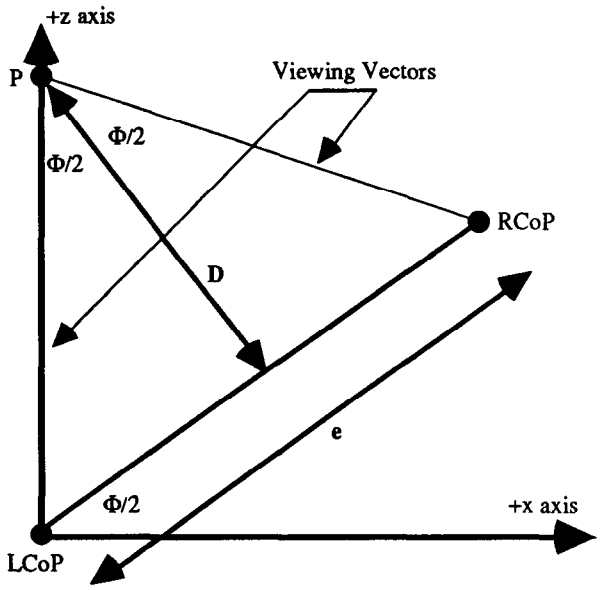


Figure 1: Displaced Parallel Projection Geometry. Rotating and translating the centers of projection - as opposed to the standard symmetrical placement about the Z axis - results in computational savings during projection.

The left-eye projection is simply a parallel projection based on the data position after the viewing transformation. The right-eye projection involves a rotation of $-\Phi/2$ about the y axis, a translation of (-e, 0, 0) to place the RCoP at the origin, and a second rotation of $-\Phi/2$ about the y axis to place P back on the z axis. Using c and s as abbreviations for $\cos(\Phi/2)$ and $\sin(\Phi/2)$ respectively, the projection matrix for the right-eye view is:

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| c 0 s 0 | 1 0 0 -e | c 0 s 0 | | $c^2-s^2$ | 0 | 2cs | -(ce) |
| 0 1 0 0 | 0 1 0 0 | 0 1 0 0 | = | 0 | 1 | 0 | 0 |
| -s 0 c 0 | 0 0 1 0 | -s 0 c 0 | | -2cs | 0 | $c^2-s^2$ | (se) |
| 0 0 0 1 | 0 0 0 1 | 0 0 0 1 | | 0 | 0 | 0 | 1 |
| rot $-\Phi/2$ | trans -e | rot $-\Phi/2$ | | projection matrix | | | |

So the view-transformed point $(x_p, y_p, z_p)$ projects to $(x_{sl}, y_{sl})$ and $(x_{sr}, y_{sr})$, where

$$x_{sl} = x_p \tag{1}$$

$$x_{sr} = x_p[c^2 - s^2] + z_p[2cs] - e\,c$$
$$= x_p \cos(\Phi) + z_p \sin(\Phi) - e \cos(\Phi/2) \tag{2}$$

$$y_{sl,\,sr} = y_p \tag{3}$$

Since e $\cos(\Phi/2)$ is a constant term, both projections can be calculated using two additions and two multiplications.

### 2.1.2 Limitations on Viewing Angle or Volume Scale in Parallel Projections

In parallel projected stereoscopic images, horizontal parallax ($x_{sr} - x_{sl}$, the distance between projections) is unbounded. Horizontal parallax should certainly be limited to the interoccular distance so that one need not become walleyed in order to view the images. Furthermore, while experienced viewers can fuse the interoccular parallax [9], most casual viewers can tolerate much less [8]. To keep horizontal parallax in a reasonable range, the z values of the data and values of the rendering constants e and $\Phi$ must be kept within a specific range.

Let $e_{min}$ and $e_{max}$ be the near (negative parallax) and far (positive parallax) limits of horizontal parallax. These can be described as a function of the physical distance from the viewer to the screen, D, expressed in screen units:

$$e_{min} = -K1\,D \tag{4}$$
$$e_{max} = K2\,D \tag{5}$$

K1 and K2 are positive constants which are generally 0.028 or less [8]. Given that a separation of $e_{min}$ should occur at the closest z point of the data and $e_{max}$ at the farthest, a range for z can be derived [1]:

$$(1 - K1/[2 \sin(\Phi/2)]) \le z \le D\,(1 + K2/[2 \sin(\Phi/2)]) \tag{6}$$

If D is fixed at some comfortable viewing distance, the range of z may be matched to the data by manipulating the viewing angle $\Phi$. Alternately, the data may be rescaled to accommodate any desired viewing angle. The value of e can be found by recognizing that the distance to point P, at which there is no parallax, is equal to the distance to the screen, D, so as can be seen in Figure 1:

$$e = 2D \tan(\Phi/2) \tag{7}$$

### 2.1. Parallel Shears

Since $\Phi$ is usually small, it has been suggested that the substitutions $\sin(\Phi) = \Phi$ and $\cos(\Phi) = 1$ could be used with little loss of accuracy [13]. Even at 6.5 degrees, more than four times the larger recommendations for e, the error is less than 0.65%. Using these simplifications reduces the x coordinate calculations as follows:

$$x_{sl} = x_p \tag{8}$$
$$x_{sr} = x_p + z_p\,\Phi - e \tag{9}$$

Both pairs of coordinates can be found in two additions and one multiplication using displaced parallel projection. The same considerations about unbounded parallax exist for shears as they do for rotations.

### 2.2 Ray-Traced Volume Rendering

In ray-traced volume rendering, such as that described by Levoy [12], rays are traced through the volume, accumulating values and opacities at intervals (regular or irregular) until an opacity limit is reached (often 1.0) or the ray exits the volume. The colors and opacities are found by trilinear interpolation from the eight nearest voxels.

Ray samples are accumulated by the use of Porter and Duff's over operation [16]:

$$S(i) \text{ over } S(j) = S(i) + (1 - \alpha(i))\,S(j) \tag{10}$$

where $S(i)$ is a color / opacity pair $[C(i), \alpha(i)]$ representing a portion of the ray closer to the viewpoint than the second pair $S(j)$.

Note that when rays are sampled in regularly spaced intervals, projection from the left-eye to the right becomes even simpler. Looking at equation *(9)*, the projection value on a ray will increase at each interval by $\Delta z\,\Phi$ (or $\Delta z \sin(\Phi)$, if rotations are used instead of shears), which is a predetermined constant. Furthermore, if the left-eye rays are processed sequentially along scan-lines, the initial right-eye projection from two consecutive left-eye rays will differ by the constant $\Delta x$ (or $\Delta x \cos(\Phi)$). On average, then, the projection value can be found with a single addition operation.

### 2.2.1 Parallel Volume Rendering

Parallel architectures are sometimes used for volume rendering because of their speed and the memory for rendering huge data sets which is not available on traditional architectures. Ma, et al. described an algorithm for parallel volume rendering using a data distributed model on the Thinking Machines CM-5 [15, 20]. In their implementation, the volume is divided among a power of two number of nodes, using a k-D tree structure with binary subdivision along successive world axes. Each subvolume is transformed to correspond to the viewing specifications and given a bounding box orthogonal to the world axes, through which rays are cast.

Rays accumulate values at regular intervals, which are dependent on volume size and not image size. While load-balancing problems remain to be addressed, the method has good characteristics for accelerated rendering. The algorithm presented here uses a slightly modified version of Ma, et al.'s code as a front-end for the stereoscopic technique. The displaced parallel

projection with shears, as described in section 2.1.3, is used. It should be noted, however, that the stereoscopic algorithm does not require a parallel renderer, and would work just as well on a serial computer.

### 2.3 Guaranteeing Correct Stereoscopic Views

If rays are allowed to traverse completely through the volume, the samples computed for the left eye are projected to the right-eye viewpoint, resulting in a correct stereoscopic view. However, a common method for accelerating the volume rendering process is early ray termination. When the accumulated ray opacity reaches 1.0, the remaining samples encountered by the ray contribute nothing to the final color.[1] Thus, it is possible to terminate the ray traversal at this point without any loss of information. Yet if the left-eye ray is terminated early, samples are ignored which might contribute to one or more right-eye rays. It is necessary to handle this case properly if a correct stereoscopic view is to be obtained.

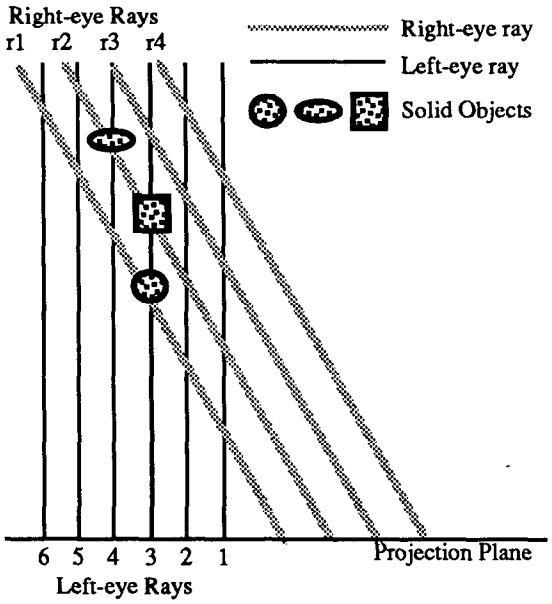


Figure 2: The round object terminates ray 3 and projects to ray r1 in the right eye. The oval object in ray 4 would project to ray r2, but that projection will not be allowed since an object (like the square) may exist past the termination point in ray 3. No right-eye projection is allowed farther to the right of the final projection of any left-eye ray already evaluated.

Figure 2 shows parallel rays from the two view points with the rotational angle greatly exaggerated for clarity. From this figure it can be seen that if rays in the left-eye view are evaluated right to left along scan-lines and front to back along the ray, the right-eye rays will accumulate values in a front to back order as well. As the left-eye rays accumulate values, the right-eye projection of the current sample is calculated, and if the appropriate right-eye ray has not reached full opacity, the sample is accumulated into that ray as well.

[1] If isosurfaces exist in the volume rendering, the opacity reaches 1.0 when a surface is encountered. Early ray termination can be thought of as the existence of occluding objects at the voxel where the opacity reaches 1.0.

Suppose, as in Figure 2, that a left-eye ray (ray 3) accumulates full opacity and terminates at the round solid object. Nothing is known about the data which may lie beyond this termination point. It is possible that samples which accumulate to a left-eye ray processed after the terminated ray (rays 4 - 6) would project into a right-eye ray whose view may be obscured by the unknown area (rays r2 - r4). To avoid this, the maximum allowable right-eye projection will be the final right-eye projection of any left-eye ray already evaluated. In this example, as ray 4 is processed, no right-eye projection will be allowed to the right of ray r1, since there may be unknown material beyond the termination of ray 3 which was never accumulated into the right-eye rays.

When all left-eye rays have been calculated, the right-eye rays are re-examined. Any right-eye ray which has not accumulated full opacity and whose last projection is not on the far side of the volume bounding box, indicates some left eye ray was terminated early. In order to generate the correct stereoscopic view, it is necessary to continue the ray traversal from the last projection into the ray. This algorithm assures that right-eye rays accumulate only those samples from the left-eye view which are guaranteed to be visible and that the right-eye view evaluates voxels only when and where it must.

It is likely that projections from the left-eye view to the right-eye view will result in the accumulation of samples from locations which would not be sampled had the right-eye view been completely rendered. Indeed, every time a sample is projected between views, it is being accumulated to the *nearest* right-eye ray. It should be remembered, however, that the angle between the two views is small (often less than two degrees), and, hence, the "error" is also quite small. It appears to be extraordinarily difficult, if not impossible, to visually notice the difference between a completely rendered image and one produced by this rendering technique.

## 3. PERFORMANCE TESTS AND DISCUSSION

A modified version of Ma, et al.'s parallel volume renderer was created which would project voxel values using the displaced parallel projection. Three sets of volumetric data were used to test the renderer: a CT scan of a human head ($128^3$ - Figure 3), a Magnetic Resonance Angiography (MRA) data set showing the vascular structure in the brain of a patient (256 X 256 X 128 - Figure 4), and a $256^3$ CFD data set computed on the CM-200 showing the onset of turbulence (Figure 5). Each set was evaluated at three different rendering resolutions (single image sizes of 64 X 64, 128 X 128, and 256 X 256) and five different node configurations (32, 64, 128, 256, and 512 nodes).

Tables 1, 2, and 3 below display the savings of the algorithm in three ways. First, the actual time savings, comparing the time to generate a full right-eye image with the fast stereoscopic algorithm time to generate the second image. This savings will be measured on a scale of 0 to 100%. The second measure is the percentage of left-eye voxels retained in the right eye view. The measure counts only those voxels which actually accumulate into right-eye rays; voxels which project to a terminated ray are ignored. The final measure is the ratio of the number of voxels

evaluated solely for the right-eye to those evaluated for the left-eye (projected to the right eye or not).[2]

Table 1: Head Data Results - Percentages

| Size | Nodes 32 | 64 | 128 | 256 | 512 |
|---|---|---|---|---|---|
| $64^2$ | | | | | |
| time savings | 74.4 | 75.2 | 73.0 | 74.3 | 73.7 |
| voxels retained from left view | 96.1 | 96.9 | 97.5 | 97.5 | 97.7 |
| voxels evaluated (right/left) | 5.1 | 4.3 | 4.0 | 4.0 | 4.3 |
| $128^2$ | | | | | |
| time savings | 78.4 | 78.6 | 78.0 | 78.4 | 79.5 |
| voxels retained from left view | 98.0 | 98.4 | 98.7 | 98.7 | 98.9 |
| voxels evaluated (right/left) | 2.6 | 2.1 | 1.8 | 1.8 | 1.8 |
| $256^2$ | | | | | |
| time savings | 80.6 | 80.4 | 79.8 | 80.5 | 81.2 |
| voxels retained from left view | 98.9 | 99.2 | 99.3 | 99.3 | 99.4 |
| voxels evaluated (right/left) | 1.3 | 1.1 | 0.9 | 0.9 | 0.8 |

Table 2: Vessel Data Results - Percentages

| Size | Nodes 32 | 64 | 128 | 256 | 512 |
|---|---|---|---|---|---|
| $64^2$ | | | | | |
| time savings | 77.4 | 76.7 | 77.1 | 79.1 | 81.0 |
| voxels retained from left view | 98.2 | 98.2 | 98.7 | 98.9 | 98.9 |
| voxels evaluated (right/left) | 2.8 | 2.8 | 2.2 | 2.5 | 2.1 |
| $128^2$ | | | | | |
| time savings | 79.8 | 79.7 | 78.7 | 79.8 | 80.2 |
| voxels retained from left view | 98.9 | 98.9 | 99.4 | 99.6 | 99.6 |
| voxels evaluated (right/left) | 1.7 | 1.7 | 1.1 | 1.0 | 1.0 |
| $256^2$ | | | | | |
| time savings | 79.9 | 80.5 | 80.2 | 80.1 | 79.9 |
| voxels retained from left view | 99.4 | 99.4 | 99.7 | 99.8 | 99.8 |
| voxels evaluated (right/left) | 0.9 | 0.9 | 0.6 | 0.4 | 0.4 |

Given that the extremely short rendering times challenged the accuracy of the timing mechanism, the differences in time savings for a particular data set are insignificant for a fixed image size but with different node configurations. The savings does increase with image resolution. As image size grows, the parallax between projected points remains the same (as it is based on viewer distance, not image size), even though the horizontal size of the image has increased. In effect, this is equivalent to keeping the image size constant and decreasing the projection distances. Since areas of terminated rays tend to appear in clusters, a smaller effective projection distance means that it is less likely that a sample from the left eye will project into a terminated ray-cluster in the right eye. More left-eye samples will then be retained in the right-eye view, and as a result, fewer samples will need to be evaluated solely for the right-eye rays. (Both phenomenon are seen in the results as image size increases.) Since the projection of left-eye samples to the right eye is a much faster operation than evaluating voxels for the right-eye, the time savings also increases with image size.

Table 3: Vorticity Data Results - Percentages

| Size | Nodes 32 | 64 | 128 | 256 | 512 |
|---|---|---|---|---|---|
| $64^2$ | | | | | |
| time savings | 77.1 | 78.5 | 77.8 | 80.7 | 81.0 |
| voxels retained from left view | 99.1 | 99.4 | 99.3 | 99.3 | 99.1 |
| voxels evaluated (right/left) | 1.2 | 1.1 | 1.3 | 1.3 | 1.9 |
| $128^2$ | | | | | |
| time savings | 80.0 | 80.4 | 79.8 | 80.1 | 80.4 |
| voxels retained from left view | 99.4 | 99.7 | 99.8 | 99.8 | 99.8 |
| voxels evaluated (right/left) | 0.8 | 0.5 | 0.5 | 0.5 | 0.6 |
| $256^2$ | | | | | |
| time savings | 80.7 | 80.8 | 80.6 | 80.7 | 80.9 |
| voxels retained from left view | 99.6 | 99.8 | 99.9 | 99.9 | 99.9 |
| voxels evaluated (right/left) | 0.6 | 0.3 | 0.2 | 0.2 | 0.2 |

The ratio of voxels retained, those samples evaluated for the left-eye view that were reused in the right-eye view, is high in all cases, better than 96%. This value will be larger if the voxels are less opaque, as this will result in fewer terminated rays in the right-eye view. The head data, which contained an isosurface, had the smallest retained voxels percentage since the rays terminated when they reached the surface. In contrast, the vessel data had at least 98.2% retention, and the vorticity data, at least 99.3%. It has already been explained that this percentage will rise with image size; it will also (generally) rise with node configuration size. When the number of nodes increases, the area of the volume which a given node is rendering will decrease. A smaller volume area lowers the probability that a ray passing through the area will terminate, so that the retained voxels percentage increases. However, there is a possibility that poor partitioning of the data will result in several dense blocks, slightly lowering the overall savings. This is seen in the 64 X 64 vorticity data, in going from 64 to 128 nodes (Table 3).

The evaluated voxel ratio of the right to left-eye is quite small. In the case of the head data with many terminated rays, the right-eye had to calculate 5% of the voxels evaluated in the left-eye. In the other data sets there are fewer terminated rays, and correspondingly fewer voxels evaluated in the right-eye, to as little as 0.2%. Figure 6 visually illustrates the limited number of samples evaluated only for the right-eye.

The evaluated voxel ratio increases inversely with image size, as explained above. A larger number of nodes will also cause the ratio to fall. As fewer rays terminate, fewer voxels will be

[2] The timing results and voxel counts used to calculate these percentages can be found in the Los Alamos technical report, LA-UR-94-1250.

evaluated only for the right-eye; there are more samples being reused from the left-eye view. Likewise, dense blocks which cause the retained voxel percentage to fall will increase the evaluated voxel percentage, as seen in Table 3.

Notice that the total percentage of voxels seen in the right-eye view (voxel retained percentage + voxel ratio percentage) is greater than 100% of the left-eye voxels. This is primarily due to the volume bounding box, which is calculated for the left-eye. As seen in Figure 2, the left-eye rays traverse this box orthogonally, while the right-eye rays pass through at an angle, which is a longer traversal. Additionally, if the volume being rendered is longer in the z dimension (after the viewing transformation) than it is in the x dimension, there will be a greater number of right-eye rays passing through the volume than left-eye rays. The total number of voxels used in the right-eye view is usually about 1% greater than those in the left-eye view, and this percentage will drop as the viewing angle decreases and with cubically-shaped volumes.

### 3.1 Stereo Pair Generation and Other Volume Rendering Techniques

It is possible to use stereo pair coherence with other volume rendering techniques. For example, the dividing cubes method [6] produces point samples of non-transparent data which are then projected to the screen. These samples can be projected to the right eye using stereoscopic techniques developed for non-volumetric ray-tracing [5], and likely achieve similar time savings of 80-90% . Some methods, like marching cubes [14], produce a polygonal isosurface which is then rendered using traditional polygon scan-conversion methods. Polygons can be clipped, back-face culled, and projected simultaneously using coherence techniques [4]. If Gouraud or Phong shading is desired, the interior of the polygons must be shaded separately, since projected polygons will have the differing widths (although identical heights). Tests with relatively large polygons (an average polygon covering 50 pixels) produced a time savings of 36%. This savings is significantly smaller than the ray-traced volume rendering, and most of the loss occurs in the interior polygon color calculation. Since polygons extracted from a surface tend to be very small, the interior of polygons will be small, and a much larger savings is expected.

Other methods for projecting voxels directly, such as splatting [22], can use the coherence to quickly find the pixels in the right-eye view affected by the voxel. Savings should be similar to that reported in this paper.

## 4. CONCLUSIONS AND FUTURE WORK

Stereoscopic images are a viable option for giving unambiguous depth information to volumetric rendering. Such images display the structure of the data without the need of continuous movement, which is usually employed in scientific visualization. A technique has been demonstrated which will generate stereoscopic pairs of volumetric data, creating the second half of the pair in a fraction of the time of generating the first half. In this implementation, the second half of the pair was generated 74 - 81% faster than full rendering, with the same quality of image. The technique is more efficient with less opaque data, with data rendered at a higher resolution, and rendering with a larger number of processors.

The technique need not be used with a parallel render. A serial render could also use this stereoscopic algorithm, much in the same way that each node of the CM-5 individually uses the algorithm on a small portion of the volume. Also, volumetric ray-tracing modifications such as adaptive screen sampling, adaptive ray sampling, ray templates, and space-leaping (described in [23]) could be implemented without affecting the reprojection technique. Using these methods may affect the time savings somewhat, since all of the left-eye voxels would be affected while only the right-eye voxels evaluated solely for that view would benefit from the faster method. This needs to be tested, but a significant retention of savings is expected.

Since the actual number of voxels rendered in the right-eye is much lower than the time savings suggests, it is likely that the time savings would be greater with optimized code. Also, when using multiple processors, a load balancing solution may increase savings, and should at the very least eliminate the minor drops in savings that were seen in Table 6, caused by poor load balancing.

## 5. ACKNOWLEDGMENTS

The authors would like to acknowledge the use of computing resources located at the Advanced Computing Laboratory (ACL) of Los Alamos National Laboratory, Los Alamos, NM 87545. This work was performed under the auspices of the United States Department of Energy.

## 6. REFERENCES

1. Adelson, Stephen, Stereoscopic Projections: Parallel Viewing Vectors, Rotations, and Shears. Los Alamos National Laboratory Technical Report LA-UR-94-0115, Los Alamos, NM, Jan., 1994.

2. Adelson, Stephen, Using stereoscopic imaging in visualization applications. In *SPIE Proceedings 2178: Visual Data Exploration and Analysis* (San Jose, California, February, 1994), 88-99.

3. Adelson, Stephen, Allen, Jeanette, Badre, Albert, Hodges, Larry and Lawrence, Andrea, Performance Comparison of Multiple Image Depth and Shape Cues. *International Journal of Human-Computer Interaction* 5, 4 (November 1993), 347-360.

4. Adelson, Stephen, Bentley, Jeffrey, Chong, In, Hodges, Larry and Winograd, Joseph, Simultaneous Generation of Stereoscopic Views. *Computer Graphics Forum* 10, 1 (March 1991), 3-10.

5. Adelson, Stephen and Hodges, Larry, Stereoscopic Ray-Tracing. *The Visual Computer* 10, 3 (December 1993), 127-144.

6. Cline, Harvey, Lorensen, William, Ludke S., Crawford, C.R. and, Teeter BC, Two Algorithms for the Three-Dimensional Reconstruction of Tomograms. *Medical Physics* 15, 3 (May / June 1988), 320-327.

7. Hsu, J., Babbs, C.F., Chelberg, D.M., Pizlo, Z. and Delp, E.J., A study of the effectiveness of stereo imaging truth or dare: Is stereo viewing really better? In *SPIE Proceedings 2177a: Stereoscopic Displays and Applications V*, (San Jose, California, February, 1994), 211-222.

8. Hodges, Larry, Time-Multiplexed Stereoscopic Computer Graphics. *IEEE Computer Graphics and Applications* 12, 2 (March 1992), 20-30.

9. Hodges, Larry, Unpublished research results relayed in personal conversation.

10. Hodges, Larry and McAllister, David, Computing Stereoscopic Views. In *Stereo Computer Graphics and Other True 3D Technologies*, ed. David McAllister, Princeton University Press, Princeton, 1993, 71-89.

11. Hodges, Larry and McWhorter, Shane, Stereoscopic Display for Design Visualization. *Image Communication* 4, 1 (November 1991), 3-13.

12. Levoy, Marc, Display of Surfaces from Volume Data. *IEEE Computer Graphics and Applications* 8, 3 (May 1988), 29-37.

13. Lipscomb, James, Three-Dimensional Cues for a Molecular Computer Graphics System. Ph.D. dissertation, Department of Computer Science, University of North Carolina, Chapel Hill, NC, 1979.

14. Lorensen, William and Cline, Harvey. Marching Cubes: A High Resolution 3D Surface Construction Algorithm. Proceedings of SIGGRAPH '87 (Anahiem, California, July 27-31, 1987). In *Computer Graphics* 21, 4 (July 1987), 163-169.

15. Ma, Kwan-Liu, Painter, James, Hansen, Charles and Krogh, Michael, "A Data Distributed, Parallel Algorithm for Ray-Traced Volume Rendering," Proceedings of the 1993 Symposium on Parallel Rendering (San Jose, California, October 25-26, 1993), special issue of *Computer Graphics*, ACM SIGGRAPH, New York, 1993.

16. Porter, Thomas and Duff, Tom Compositing Digital Images. Proceedings of SIGGRAPH '84 (Minneapolis, Minnesota, July 23-27, 1984). In *Computer Graphics* 18, 3 (July 1984), 253-259.

17. Reinhart, William, Depth Cueing for Visual Search and Cursor Positioning. In *SPIE Proceedings 1457: Stereoscopic Displays and Applications II*, (San Jose, California, February, 1991), 221-232.

18. Reinhart, William, Beaton, Robert and Snyder, Harry, Comparison of Depth Cues for Relative Depth Judgments. In *SPIE Proceedings 1256: Stereoscopic Displays and Applications*, (San Jose, California, February, 1990), 12-21.

19. Sollenberger, R.L. and Milgram, P., "A Comparative Study of Rotational and Stereoscopic Computer Graphics Depth Cues." Proceedings of the Human Factors Society 35th Annual Meeting, (October 1991), 1452-1456.

20. Thinking Machines, The Connection Machine CM-5 Technical Summary (1991).

21. Ware, Colin, Arthur, Kevin and Booth, Kellogg, Fish Tank Virtual Reality. In *INTERCHI '93*, (April, 1993), 37-42.

22. Westover, Lee, Footprint Evaluation for Volume Rendering. Proceedings of SIGGRAPH '90 (Dallas, Texas, August 6-10, 1990). In *Computer Graphics* 24, 4 (August 1990), 367-376.

23. Yagel, Roni, Volume Viewing: State of the Art Survey. In *ACM SIGGRAPH '93 Course Notes #41: Volume Visualization*, August, 1993, pp. 102-129.

24. Yeh, Yei-Yu and Silverstein, Louis, Visual Performance with Monoscopic and Stereoscopic Presentation of Identical Three-Dimensional Visual Tasks. In *1990 SID International Symposium Digest of Technical Papers* (Las Vegas, Nevada, May, 1990), 359-362.


Figure 3: Stereo pair of the head data. The right eye image was generated using the fast stereo imaging technique.

Figure 4: Stereo pair of the blood vessel data. The right eye image was generated using the fast stereo imaging technique

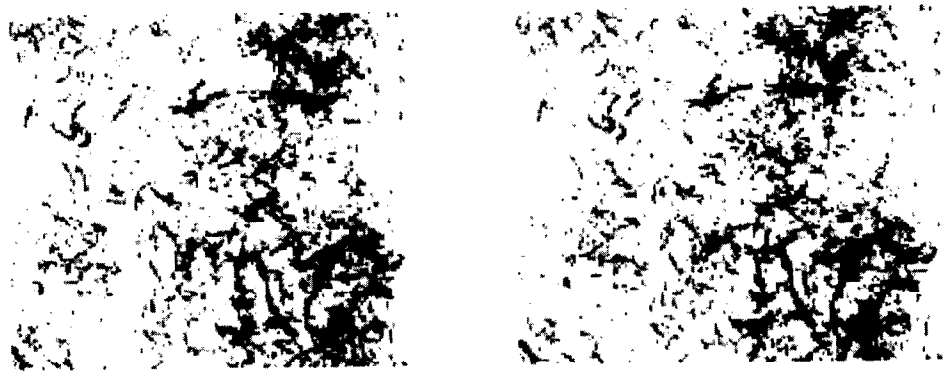

Figure 5: Stereo pair of the vorticity data. The right eye image was generated using the fast stereo imaging technique.

Figure 6: Right-eye view of the vorticity data, with samples evaluated only for the right-eye illuminated.

Figure 3: Stereo pair of the head data. The right eye image was generated using the fast stereo imaging technique.

Figure 4: Stereo pair of the blood vessel data. The right eye image was generated using the fast stereo technique.

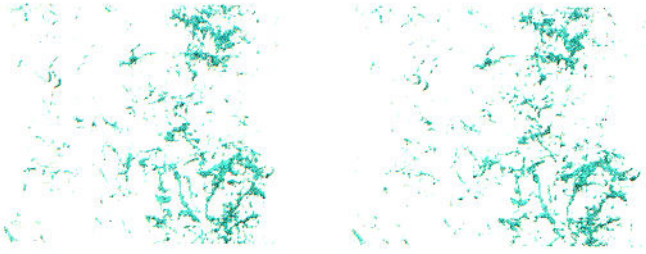

Figure 5: Stereo pair of the vorticity data. The right eye image was generated using the fast stereo imaging technique.

Figure 6: Right-eye view of the vorticity data, with samples evaluated only for the right eye illuminated.

Adelson and Hansen, "Fast Stereoscopic Images with Ray-Traced Volume Rendering"