Fast Volume Carving

Soon Hyoung Pyo

Virtual Reality Department, Computer·Software Laboratories, ETRI, Taejon, Korea.

Yeong Gil Shin

Department of Computer Science and Engineering, Seoul National University, Seoul, Korea.

Abstract

In recent years, the volumetric scene representation has been widely used in the fields of computer graphics and computer vision. In computer graphics, the representation is used to visualize and reconstruct 3D internal structures of 2D human-body medical data obtained from Magnetic Resonance Imaging (MRI) or Computed Tomography (CT). On the other hand, the volumetric representation is used to model 3D objects from a sequence of 2D images in computer vision. In both cases, eliminating some unnecessary part of the volumetric data is sometimes needed. For example, we need to remove skin or tissue which hides the target organ of human in volume rendering, or empty region of 3D objects being modeled in image-based modeling. In this paper, we propose a fast carving algorithm which can be applied in volume rendering and image-based modeling. The memory usage is minimized by using 1-bit field of a voxel to represent whether the voxel is carved or not. Our experiments show that we can carve out any part of the 512^3 *volume data in about a second. The carving speed is not affected by the complexity of objects.*

Categories and Subject Descriptors: I.3.8 [Computer Graphics]: Applications

1. Introduction

The volumetric scene representation is most widely used in 3D medical imaging and image-based modeling. In medical imaging, the volume data is represented as a sequence of 2D slices obtained from the medical scanners such as Computed Tomography (CT), Magnetic Resonance Imaging (MRI), or the ultrasound system. Those images are aligned and interpolated to form a regular volume data. Each voxel has a density value at the point. Most previous research work is focused on efficient methods of visualizing the volume data and applying them to practical diagnosis, rather than on manipulating and editing the volume data. On the contrary, research in image-based modeling is concerned primarily on modeling and manipulating the volume. Here the volume data is used as a temporary data structure to model the 3D objects, and the model is rendered after converted to a polygonal model. Since only the validity of a voxel may be enough information for the voxel, the volume data used in this area has relatively few information. For each voxel, the validity of a voxel is decided by projecting the voxel to each image and checking whether the corresponding pixel is inside the object's boundary. This decision process is the essential part of image-based modeling which uses the volumetric representation.

In this paper, we propose a fast new volume-carving algorithm. The volume-carving method can be used to eliminate invalid voxels from volume data during the image-based modeling process. Moreover, this method is a useful tool for volume rendering. In volume rendering, a mapping process is performed prior to rendering in order to convert the acquired data to those containing density and more. This process is called the classification. For example, in the case of the CT, each voxel of acquired data contains a value from –1022 to 3064, and we should map the values to the density values between 0 and 1. By varying the mapping function and the applied range, you can visualize various parts of the human body. However, users sometimes want to see the colon which is occluded by other organs which have indistinguishable CT numbers from colon. Therefore, to have a clear view of any internal organ, we need to get rid of the portions of the volume data. Existing tools such as VOI (Volume of Interest) and a cutting plane are not appropriate to be used for this purpose; they have limitations on their carving capability to achieve certain shapes. Our volume carving method puts no restriction on the shape of the carved region, giving the user a capability to carve out arbitrary regions of the volume data.

Our new algorithm uses shear-warp factorization of viewing transform to carve out the volume data. First, we create a mask image either from the user-specified carving line used in volume rendering or from the input image used in image-based modeling. Then, using the warp transform obtained from the shear-warp factorization process, we warp the mask image to obtain an intermediate image. This intermediate image is aligned with the axis of the volume by the warping. Now, pair-wise AND-operations are performed between the intermediate image and each slices of the volume data. At this point, we shift the slices according to the shear transform obtained from the shear-warp factorization. In fact, this is a reverse process of the volume rendering which uses shear-warp factorization of the viewing transform.

Our prototype implementation running on a Pentium IV-processor computer can carve out any part of the 512^3 volume data in less than a second. This is faster than the methods which use octree to accelerate the modeling process. Furthermore, because each voxel contains only valid information, it occupies only one bit of the main memory. For example, the algorithm uses only about 16M bytes for the 512^3 volume data. The efficiency of this algorithm shows some potential to be useful in modeling higher resolution volumetric objects in image-based modeling. This is important because the resolution of the volume data affects the quality of the final 3D model.

The paper is organized as follows. We begin, in Section 2, by describing some related work. In Section 3, we present the shear-warp factorization of the viewing transform in order to explain our algorithm in detail. In Section 4, we present the fast volume-carving algorithm using the shear-warp factorization of the viewing transform described in Section 3. Section 5 provides the experimental results. We draw a conclusion of our work in Section 6.

2. Related Work

The volumetric scene representation is widely used in many applications. In particular, researches in medical imaging are primarily concerned with efficient rendering of the volume data obtained from medical scanners. But relatively few researches on manipulating the volume data have been carried out. We will now present some results on editing the volume data.

The representative application of the volume editing is volume sculpting[1, 2, 5, 6, 15]. Volume sculpting is an interactive process of making a volumetric model of object from cubic volume data with an optional haptic interaction. Interactive modeling and editing of the volume data using a discretized tool was introduced by Galyean and Hughes[6]. Wang and Kaufman provided various voxel-based tools to sculpt a volume and showed significant results[15]. Baerenzen also presented a volume sculpting system where volume data was sculpted by adding and subtracting solids, and with the tools that are based on a spray-can metaphor[2]. He used an Octree data structure to represent the volume. Most recently, Ferley *et al.* introduced a metaphor of sculpting based on multi-resolution volumetric representation to sculpt the volume in various scale[5]. These works have primarily focused on developing interactive modeling technique with the volume and the metaphors. Because they are using haptic devices to sculpt the volume data, accurate and subtle editing is possible. But these methods are not appropriate for modeling from images and fast deleting mass portion of the volume data.

In computer vision, the volumetric scene representation has been used to model a 3D object from the silhouette information available from the sequence of 2D images. The main tool for the modeling in this method, called "*Shape-from-Silhouette*", is the volume intersection technique. The volume intersection technique is finding the intersection from two or more conic volumes created from the silhouette in the image and the camera center. The infinite number of volume intersections creates an object, which encloses the real object. This object is called *visual hull* of the real object[8]. Recovering the volumetric description of an object from the multiple silhouette using the volume intersection approach was first proposed by Martin and Aggarwal[10]. And most other works are using this approach.

The easiest implementation of the volume

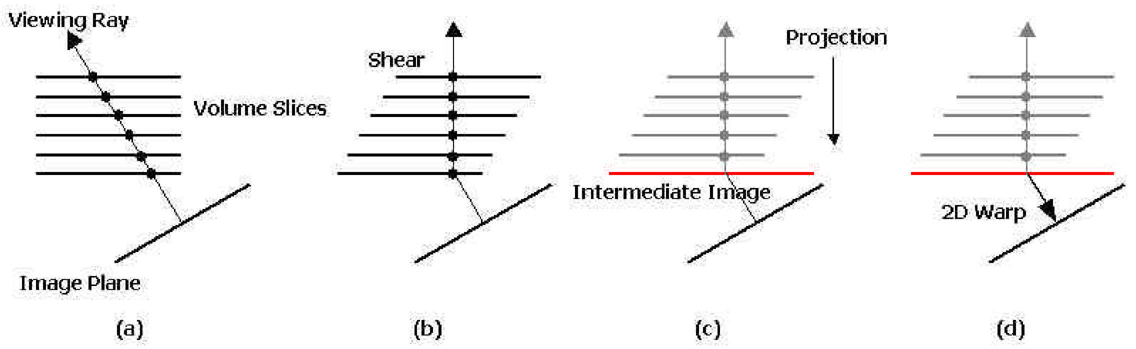


Figure 1: *The overall steps for volume rendering using the shear-warp factorization of the parallel viewing transform matrix: (a) the 2D view of the volume slices and the image plane, (b) the shear transform matrix shears the volume slices so that viewing rays are parallel with the z-axis, (c) an intermediate image is created by projecting all the volume slices, (d) warping the intermediate image with the warp transform matrix makes a final volume rendered image.*

intersection algorithm is simply to project each voxel to the silhouette image and to find out whether the projected position is inside of the silhouette or not. However, this process is too slow to be applied to any interactive application. Efficient methods for constructing an octree to represent the volume space is proposed in some papers[11, 14]. Potmesil represented a conic volume as an octree and proposed an algorithm to generate the octree of the intersection volume from them[11]. Szeliski also presented a method for reconstructing an octree from an image sequence[14]. An octree of an object is reconstructed while the object is being rotated on a turn-table.

There is another approach that uses volumetric representation to reconstruct a 3D model from the image sequences. Seitz and Dyer proposed a method, called *voxel coloring*, which uses color consistency among the images to reconstruct a volumetric model[15]. This method assigns a color value for each voxel and is shown to provide good result. But, there are some restriction on the location and the direction of the camera due to the visibility problem. Later, Culbertson extended this method to the general cases[4]. However, all these methods are computationally more expensive in reconstructing a 3D model from the image sequences.

In the next section, we present the shear-warp factorization in detail. This algorithm is essential to the development of the key idea of the proposed volume carving method.

3. The Shear-Warp Factorization

The shear-warp factorization is the fastest technique available in volume rendering as far as pure software implementation is concerned. This technique factorizes an affine transform matrix to a multiplication of a shear transform matrix and a warp transform matrix. Although this can be also applied to a perspective transform matrix, in this paper we will not consider applying the factorization to that case. Instead, we only implement the case of using a parallel viewing transform matrix.

First, the shear-warp factorization is applied to rotation matrices by Cameron and Undrill[3], and by Schröder and Stoll[12]. Later, Lacroute and Levoy applied the shear-warp factorization to a parallel viewing transform matrix and extended it to a perspective viewing transform matrix[7]. They added a permutation matrix *P* to transpose the coordinate system in order to make the z-axis the principal viewing axis. The matrix *P* is determined by calculating which axis is most close to the viewing direction of the camera. The matrix form of the shear-warp factorization is as follows:

$$M_{view} = P \cdot S \cdot M_{warp}$$

S shears the volume slices so that all viewing rays are parallel to the third coordinate axis, that is, z-axis. For a general parallel projection, *S* has the form of a shear perpendicular to the z-axis:

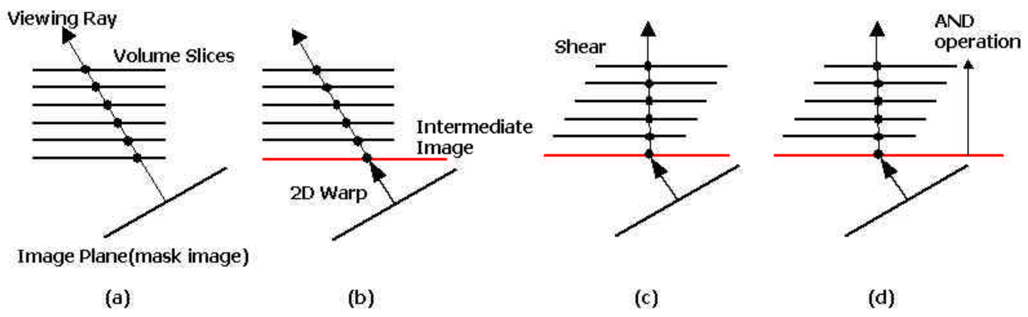


Figure 2: *The overall steps for volume carving using the shear-warp factorization of the parallel viewing transform matrix: (a) the 2D view of the volume slices and the image plane, (b) warp a mask image to a intermediate image using the warp transform matrix, (c) the shear transform matrix shears the volume slices so that viewing rays are parallel with the z-axis, (d) perform the AND operations between each volume slice and the intermediate image.*

$$S_{parallel} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ s_x & s_y & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

S_x and S_y can be obtained from the viewing matrix. Now, we can compute the M_{warp} matrix from the *P*, *S*, and M_{view} matrices.

The above-mentioned steps of the shear-warp factorization process can be summarized as follows[7]:

1. Find the principal axis and choose the corresponding permutation matrix P. Compute the permuted view matrix from M_{view} and P.
3. Compute the shear coefficients from the permuted view matrix.
4. Compute the shear matrix S and the warp matrix M_{warp}.

Using these transformations, the volume rendering can be accomplished fast[7]. Volume slices are first transformed by the shear transform matrix, which makes the viewing ray to be aligned with the z-axis. Then, volume slices are projected to the intermediate image to form a distorted image. The slice, which is near to the intermediate image, is projected first. The projection means the accumulation of the density and color value to the intermediate image. Finally, the intermediate image is transformed to the result image by the warp transform matrix. Figure 1 shows the overall process.

4. Volume Carving Algorithm

In this section, we present a new volume-carving algorithm which uses the shear-warp factorization described in the section 3. In volume rendering, the shear-warp factorization is used to make an image from the volume slices. But, we modify the volume slices from a known mask image using the shear-warp factorization. In other words, the main difference between the volume rendering and the carving algorithm lies in the order of applications of the factorized transforms.

As an input, we use a viewing transform of the camera and the corresponding image. In the case of the volume rendering application, the input image contains the user-specified carving line drawn on the rendered volume image. We can use a real or synthesized image of any object to be modeled by image-based modeling. From this input image, a mask image is created for the volume carving. The pixel values of the mask image are 0 for the image parts that should be carved and 1 for the other parts. The memory size occupied by a pixel in the mask image is the same as that of a voxel, because we perform boolean operations between them. In our implementation, 1 bit is used for both the pixel and voxel to reduce the memory usage.

When a mask image is prepared, the viewing transform matrix is factorized to a shear transform matrix and a warp transform matrix. The procedure for the factorization is the same as that of Lacroute *et al.* [7]. Now, we transform the mask image into an intermediate image using the warp transform matrix and the appropriate permutation matrix. By this transformation, the warped mask image becomes aligned with the volume slices. Next, the application of the shear transform matrix makes the viewing ray of the camera parallel to the z-axis. This means that each volume slice is sheared to make the viewing ray be parallel to the z-axis. Figure 2 shows these steps in a sequential manner.

Finally, we carve out the volume data by performing the AND-operations between the intermediate mask image and each volume slice. We implemented this algorithm using Intel Image Processing Library. Unlike other algorithms for carving a volume, our algorithm does not include a verifying step to check whether the projected image of a voxel or a group of the voxels is in the object's boundary in the reference image. Regardless of the contents in the reference image, only a single AND-operation is performed for a voxel. The omission of the verification makes the proposed algorithm be more efficient. Of course, the algorithm can be made faster by using an additional data structure like a run-length encoded volume to skip the AND-operations for the already carved voxels.

In summary, the volume-carving algorithm works as follows:

1. From the input image generate a mask image.
2. Factorize the viewing transform to a permutation matrix, a shear transform matrix, and a warp transform matrix.
3. Transform the mask image to the intermediate image by using the warp transform matrix.
4. Transform each volume slice by using the shear transform matrix.
5. Perform the AND operations between the volume slice and the intermediate mask image.

In the case of the image-based rendering, a final volumetric representation of an object is obtained through several carving processes of the volume data using different input images.

5. Experimental Results

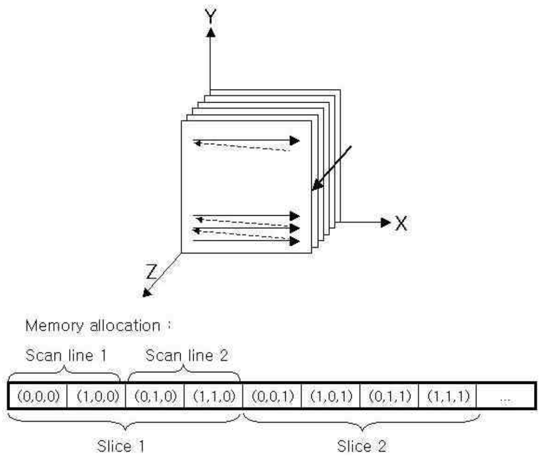
In this section, we present experimental results and give a comparison of our method with other approaches in terms of time performance. We also applied our method to both the volume rendering and the image-based modeling.

We tested two other methods in order to compare their performance with our suggested algorithm. One is the voxel projection method, which simply projects each voxel to the reference image. If the value at the projected position in the reference image is valid, the voxel is set to valid. Otherwise, the voxel is carved out. This method is very simple, but overall it shows a very poor performance. The other approach we have tested is an octree-based one. This method shows much faster processing time, because changes in the octree become rare as the carving processes are carried on for subsequent images. However, it has a relatively long initial processing time in constructing an object-shaped octree from a cube. Thus, both of these methods are not appropriate to be used for the volume rendering applications, where fast response time is necessary. Figure 3 shows the comparison of the three methods.

	1st	2nd	3rd	4th	5th	Avg.
Voxel Projection	37.20	37.47	37.20	37.65	37.49	37.40
Octree-based	11.24	4.71	2.23	1.85	1.97	4.40
Proposed Method	0.17	0.11	1.83	1.83	1.85	1.16

Figure 3: *The comparison of volume-carving time performance among voxel projection method, octree-based approach, and the proposed algorithm*

In our proposed method, the times taken for carving the second and third image are 0.17 seconds and 1.83 seconds respectively. The difference in the carving time is due to the order in which volume data is stored in the main memory. In general, the volume data is stored in the main memory as shown in Figure 4. Thus, if the viewing direction of the camera is close to the z-axis, we can perform the AND-operations slice by slice. Otherwise, another trick should be used because the logical volume slice is not stored at the consecutive memory location. We used a temporary volume data to speed up the operation. But it produces temporal delay in the results. Figure 5 shows the result of the proposed volume-carving algorithm according to the viewing



direction for a 256^3 and a 512^3 volume data. In the best case we can carve out the 512^3 volume data in

Figure 4: *The memory allocation of voxels in the main memory*

less than a second. In this way, if the interactive display of carved result is not required, we can speed up the carving by using two more volumes for the cases in which camera-viewing is aligned for the x-axis and the y-axis, respectively. In that case, after several carving processes of the volume, the results of the three volume data should be merged to generate the final result of the carving.

	256^3	512^3
X - Axis	1830ms	15702ms
Y - Axis	1844ms	15753ms
Z - Axis	147ms	871ms

Figure 5: *The performance of the algorithm according to the principal axis*

Figure 6 and Figure 7 show the application of our proposed algorithm to the volume rendering and to the image-based modeling, respectively. In Figure 6, we carve out the skin part of the volume data to be able to view clearly the volume rendered colon. If a user specifies a region to be eliminated using a provided drawing tool, the application can generate a mask image and subsequently carves the volume data. Because our algorithm is based on an image-oriented approach, an application can provide the undo/redo functions using the image lists.

In Figure 7, we used the algorithm to model a 3D object from the silhouette in the reference image. Figure 7 (b) is the result image of a model created from the input images shown in Figure 7 (a). The result image is rendered by the technique called binary volume rendering. The more input images are used, the finer 3D model is created. Using the Marching Cube

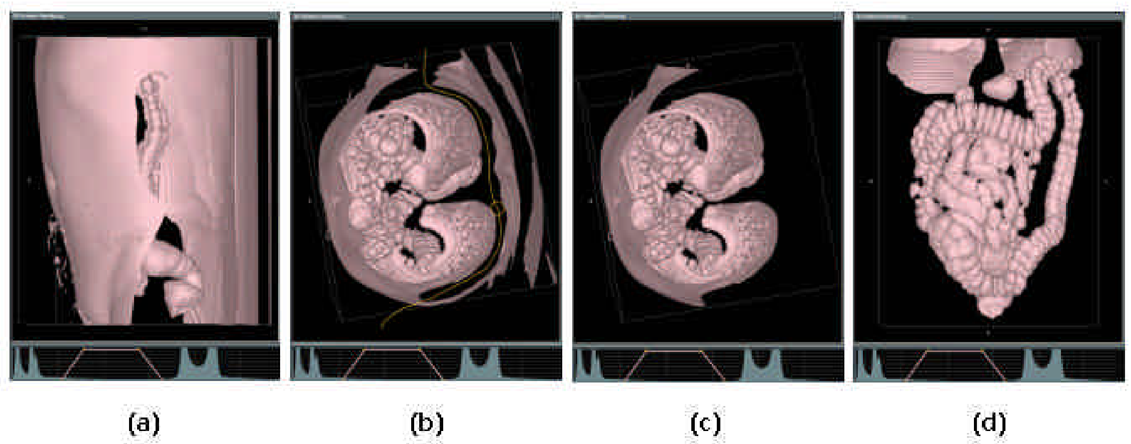


Figure 6: *The application of our algorithm to the volume rendering: (a) the colon is hidden by the skin, (b) user-specified carving line on the volume rendered image, (c) after the first carving operation, (d) after several carving operation we can see the colon cleanly*

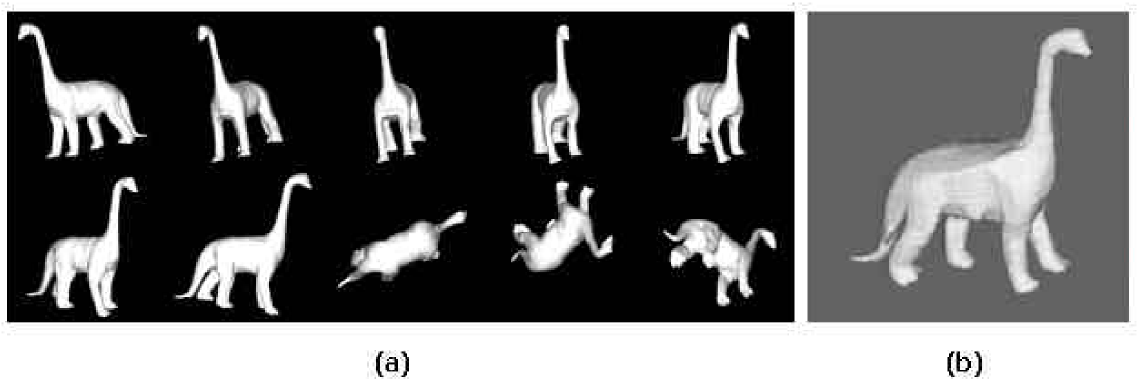


Figure 7: *The application of our algorithm to the image-based modeling: (a) the 10 reference images, (b) the result image created from the 10 reference images.*

algorithm, the model can also be converted to a polygonal model[9].

6. Conclusion

In this paper, we proposed an efficient volume-carving algorithm. The algorithm uses the shear-warp factorization of the viewing transform to carve out a part of the volume data. The experimental result shows that the method is faster than other methods such as an octree-base approach. We also presented the result of applying the proposed algorithm to volume rendering and image-based modeling.

Currently, we are working on extending the algorithm to utilize a perspective viewing transform. This possible extension is already presented by Lacroute *et al.*[7] This method was shown to be computationally more expensive due to additional steps required for scaling each volume slice. In this paper, we are not using the status information of the current volume data. We expect that skipping the AND-operations for the already carved voxel will help improve the performance. We are also planning to apply this algorithm to the image-based modeling by using real images or video sequences.

References

1. R.S. Avila and L.M. Sobierajski, "A haptic interaction method for volume visualization", In *Proceedings of IEEE Visualization'96*, October 1996, pp. 197-204.
2. Andreas Baerentzen, "Octree-based volume sculpting", In *Proceedings of IEEE Visualization '98*, 1998.
3. Cameron, G. G. and P. E. Undrill, "Rendering volumetric medical image data on a SIMD-architecture computer", In *Proceedings of the Third Eurographics Workshop on Rendering*, 135-145, Bristol, UK, May 1992.
4. W. B. Culbertson, T. Malzbender, and G. Slabaugh, "Generalized voxel coloring", In *Proceedings of International Workshop on Vision Algorithms*, Volume 1883 of *Lecture Note in Computer Science*, pp. 100-115, Springer-Verlag, 2000.
5. Eric Ferley, Marie-Paule Cani, Jean-Dominique Gascuel, "Resolution adaptive volume sculpting", *Graphical Models*, March 2002.
6. T.A. Galyean and J.F. Hughes, "Sculpting : An interactive volumetric modeling technique", In *Proceedings of SIGGRAPH'91*, July 1991.
7. Philippe Lacroute and Marc Levoy, "Fast Volume Rendering Using a Shear-Warp Factorization of the Viewing Transform", In *Proceedings of SIGGRAPH '94*, Orlando, Florida, July, 1994, pp. 451-458.
8. Aldo Laurentini, "The visual hull concept for silhouette-based image understanding", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 16, No. 2, pp. 150-162, February 1994.
9. Lorensen W.E. and Cline H.E., "Marching Cubes: A High-Resolution 3D Surface Construction Algorithm", In *Proceedings of SIGGRAPH '87*, pp. 163-169, July 1987.
10. W. N. Martin and J. K. Aggarwal, "Volumetric description of objects from multiple views", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. PAMI-5, pp. 150-158, 1994.

11. M. Potmesil, “Generating octree models of 3D objects from their silhouettes in a sequence of images”, *CVGIP: Image Understanding,* Vol. 40, pp. 1-29, 1987.
12. Schröder, Peter and Gordon Stoll, “Data parallel volume rendering as line drawing”, In *Proceedings of the 1992 Workshop on Volume Visualization*, 25-32, Boston, October 1992.
13. S. Seitz, Charles R. Dyer, “Photorealistic scene reconstruction by voxel coloring”, In *Proceedings of Computer Vision and Pattern Recognition Conference*, pp. 1067-1073, 1997.
14. R. Szeliski, “Rapid octree construction from image sequences”, *CVGIP: Image Understanding,* Vol. 58, No. 1, pp. 23-32, July 1993.
15. S. Wang and A. Kaufman, “Volume sculpting”, In *Proceedings of Symposium on Interactive 3D Graphics*, ACM Press, April 1995, pp. 151-156.