Real-Time Active 3D Shape Reconstruction for 3D Video

X. Wu and T. Matsuyama
Department of Intelligent Science and Technology,
Graduate School of Informatics, Kyoto University

Abstract

In this paper, we focus on the real-time 3D shape reconstruction of an object moving in a wide area(e.g. a dancing human). To efficiently record dynamically changing 3D object shape, the real-time 3D shape reconstruction and active tracking are required. For the real-time 3D shape reconstruction, we propose a parallel plane-based volume intersection method. By implementing the algorithm on a PC cluster system, we have succeeded in real-time 3D shape reconstruction. Then to get the 3D shape of a moving object in a wide space, we introduce active object tracking by multiple cameras. To realize the active 3D shape reconstruction, we augmented the volume intersection method so that it can be applied to those images captured by active cameras.

1. Introduction

3D video is the ultimate image medium recording dynamic visual events in the real world as is: time varying 3D object shape with high fidelity surface properties (i.e. color and texture)[1]. Its applications cover wide varieties of personal and social human activities: entertainment (e.g. 3D game and 3D TV), education (e.g. 3D animal picture books), sports (e.g. sport performance analysis), medicine (e.g. 3D surgery monitoring), culture (e.g. 3D archive of traditional dance) and so on.

In recent years, several research groups developed real-time 3D shape reconstruction systems for 3D video and have opened up the new world of image media [1] [2] [3] [4] [5] [6]. All these systems focus on capturing human body actions and share a group of distributed video cameras for real-time synchronized multi-viewpoint action observation. While the real-timeness of the earlier systems[5] [6] was confined to the synchronized multi-viewpoint video observation alone, the parallel volume intersection on a PC cluster has enabled the real-time 3D shape reconstruction [1] [2] [3] [4].

This paper focuses on the real-time 3D shape acquisition of an object moving in a wide area. We classify the 3D shape reconstruction into the following two levels:

- **Level 1:** Real-time 3D shape reconstruction without camera action.

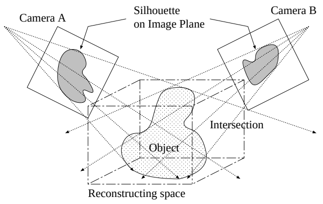


Figure 1. Silhouette Volume Intersection.

- **Level 2:** Real-time 3D shape reconstruction *with* camera action for tracking a moving object.

For the level one, we propose a volume intersection method based on the parallel plane-to-plane projection . Experimental results are shown to prove the efficiency of our proposed method.

For the level two, since the camera action causes the dynamic changes of the camera locations(i.e. the positions and the directions of the cameras), we have to extend the volume intersection method so that it can be applied to varieties of camera arrangements.

2. Real-Time 3D Shape Reconstruction

Silhouette Volume Intersection [9] [10] [14] [15] [17] [18] is the most popular idea for computing 3D shape of the object (Figure 1). This idea is based on *silhouette constraints* that each 2D silhouette of an object constrains the object inside the frustum produced by back-projecting the silhouette from the corresponding viewpoint. Therefore, by intersecting the frusta for all silhouettes, an approximation of the object volume is obtained. This is called *visual hull* [13] which constrains the object in its inside. Recently, this idea is further extended using photometric information to reconstruct more accurate shapes [12].

In the volume intersection method[13], the perspective projection process requires very expensive computation; it involves a considerable amount of arithmetic operations.

To accelerate the computation, we first developed the plane-based volume intersection method, where the 3D voxel space is partitioned into a group of parallel planes and the cross-section of the 3D object volume on each plane is reconstructed. Secondly, we devised the Plane-to-Plane Perspective Projection(**PPPP**) algorithm to realize efficient

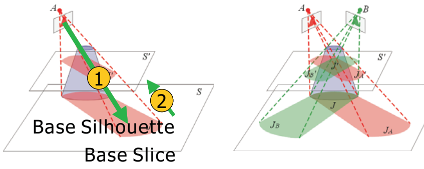


Figure 2. Plane-based Volume Intersection Method

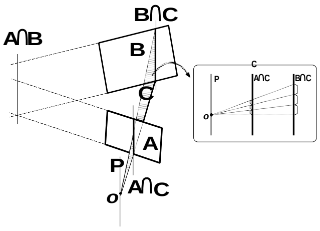


Figure 3. Linear PPPP algorithm

plane-to-plane projection computation. And thirdly, to realize real-time processing, we implemented parallel pipeline processing on a PC cluster system. In what follows, we describe these methods in details.

2.1. Plane-based Volume Intersection Method

Figure 2 illustrates the plane-based volume intersection method. By partitioning the 3D space into a group of parallel planes, the 3D shape of the object can be reconstructed by calculating a group of 2D cross-sections of the object on the planes. The cross-section on each plane can be obtained by back-projecting the multi-view silhouettes to the plane and calculating 2D intersection among the projected silhouettes (Figure 2). It should be noted that this plane-to-plane back-projection (*homography*) [16] is computationally less expensive than general 3D perspective projection;

To accelerate the plane-to-plane projection computation further, we developed the following algorithm.

2.2. Accelerated Plane-to-Plane Perspective Projection Algorithm

Based on the geometric relations between a pair of planes involved in the projection, the acceleration of the plane-to-plane perspective projection can be done in the following two ways:

1. For planes which are not parallel, we devised the linear PPPP algorithm (Figure 3).
2. For parallel planes, we apply the plane-wise PPPP algorithm.

Both algorithms consist of simple linear computations, which can be executed by popular graphic hardwares.

Linear Plane-to-Plane Perspective Projection

In Figure 3, we want to map a silhouette on plane A onto B, where A and B are not parallel. $A \bigcap B$ denotes the intersection line between the planes and O the center of perspective projection. Let P denote the line that is parallel to $A \bigcap B$ and passing O. Then, take any plane including P (C in Figure 3), the image data on the intersection line $A \bigcap C$ is projected onto $B \bigcap C$. As shown in the right part of Figure 3), this linear (i.e. line-based) perspective projection can be computed just by scaling operation, since $A \bigcap C$ and $B \bigcap C$ are parallel to each other. This means 4 additions are enough to compute the projection of a point. But, there are initialization overheads to compute (1) starting point pair and (2) scaling coefficients. These overheads are equivalent to computing two projections, two vector subtractions, two scalar divisions per each line pair.

Plane-wise Plane-to-Plane Perspective Projection

The projection between two parallel planes is simplified to 2D isotropic scaling and translation, which requires **2** additions and **2** multiplications per point.

With these two kinds of accelerated PPPP algorithms, the back-projection process in the plane-based volume intersection method can be divided into the following two stages:(Figure 2) (1) First, with the linear PPPP, back-project an object silhouette on the image plane onto the base plane, which is one of the parallel planes decomposing the space. (2) Then, with the plane-wise PPPP, project the base silhouette onto each of the parallel planes.

To realize real-time 3D shape reconstruction, we next implement a parallel pipeline processing method for the above mentioned plane-based volume intersection.

2.3. Parallelized Volume Intersection Method

According to what has been described, the process of the plane-based volume intersection method can be divided into following stages:

1. Back-projection
2. Silhouette intersection on each plane

To make this processing parallel on our PC cluster, we observe

- Since the back-projection is closely connected with image capturing and silhouette extraction processes, it should be executed on the same PC that captures an image.
- Since the silhouette intersection on each plane can be done independently of that on the others, we partition a set of parallel planes into a group of subsets and assign a subset to each PC, which computes the silhouette intersection on those planes included in its assigned subset.
- To realize the above parallel silhouette intersection, we have to make each PC have a full set of multi-view silhouettes. That is, after computing its own base plane silhouette, each PC broadcasts that data to all the other PCs. As will be proved later, this broadcasting does not introduce large overhead, because the data size transmitted is small (i.e. 2D bit image data representing the base plane silhouette) and the network speed

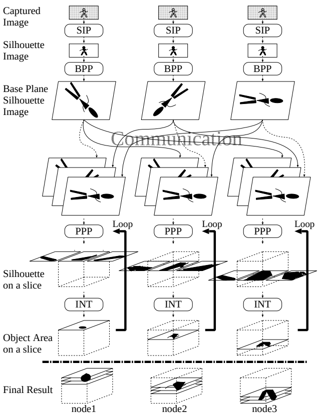


Figure 4. Processing flow of the parallel pipelined 3D shape reconstruction.

is very high. Note that this silhouette duplication enables completely parallel silhouette intersection on the planes without any overhead.

Figure 4 illustrates the processing flow of the parallel pipelined 3D shape reconstruction. It consists of the following five stages:

1. *Image Capture :* Triggered by a capturing command, each PC with a camera captures a video frame (Figure 4 top row).

2. *Silhouette Extraction :* Each PC with a camera extracts an object silhouette from the video frame (Figure 4 second top row).

3. *Projection to the Base-Plane :* Each PC with a camera projects the silhouette onto the common base-plane in the 3D space (Figure 4 third top row).

4. *Base-Plane Silhouette Duplication :* All base-plane silhouettes are duplicated across all PCs over the network so that each PC has the full set of all base-plane silhouettes (Figure 4 forth top row). Note that the data are distributed over all PCs (i.e. with and without cameras) in the system.

5. *Object Cross Section Computation :* Each PC computes object cross sections on specified parallel planes in parallel (Figure 4 three bottom rows).

In addition to the above parallel processing, we introduced a pipeline processing on each PC: 5 stages (corresponding to the 5 steps above) for PC with a camera and 2 stages (the step 4 and 5) for PC without a camera. In this pipeline processing, each stage is implemented as a concurrent process and processes data independently of the other stages. Note that since a process on the pipeline should be synchronized with its preceding and succeeding processes and moreover the stage 5 for the silhouette intersection cannot be executed before all silhouette data are prepared, the output rate, the rate of the 3D shape reconstruction, is limited to the rate of the slowest stage.

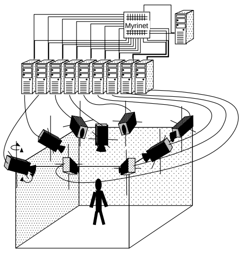


Figure 5. PC cluster for real-time active dynamic 3D object shape reconstruction system.

Figure 6. Multi-View Image Samples Captured by the PC cluster system.

2.4. Performance Evaluation

In the experiments of the real-time 3D volume reconstruction, we used 6 digital IEEE1394 cameras placed at the ceiling for capturing multi-view video data of a dancing human(like Figure 5, and Figure 6 shows the input samples). Each PC has dual PentiumIII 1Ghz installed and connected by a myrinet network. We will discuss their synchronization method later. The size of input image is 640×480 pixels and we measured the time taken to reconstruct one 3D shape in the voxel size of 2cm$\times$ 2cm$\times$ 2cm contained in a space of 2m $\times$ 2m $\times$ 2m.

In the first experiment, we analyzed processing time spent at each pipeline stage by using 6 - 10 PCs for computation. Figure 7 shows the average computation time [1] spent at each pipeline stage. Note that the image capturing stage is not taken into account in this experiment, which will be discussed later.

From this figure, we can observe the followings:

- The computation time for the *Projection to the Base-Plane* stage is about 18ms, which proves the accelerated PPPP algorithm is very efficient.

- With 6 PCs (i.e. with no PCs without cameras), the bottleneck for real-time 3D shape reconstruction rests at the *Object Cross Section Computation* stage, since this stage consumes the longest computation time (i.e. about 40ms). By increasing the number of PCs, the time decreases considerably. This proves the proposed parallelization method is effective.

In the second experiment, we measured the total throughput of the system including the image capturing process by changing the numbers of cameras and PCs. Figure 8 shows the throughput[2]to reconstruct one 3D shape.

[1] For each stage, we calculated the average computation time of 100 video frames on each PC. The time shown in the graph is the average time for all PCs.

[2] The time shown in the graph is the average throughput for 100 frames.

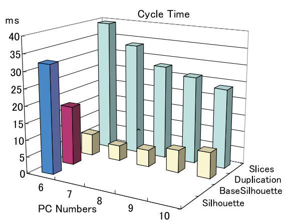
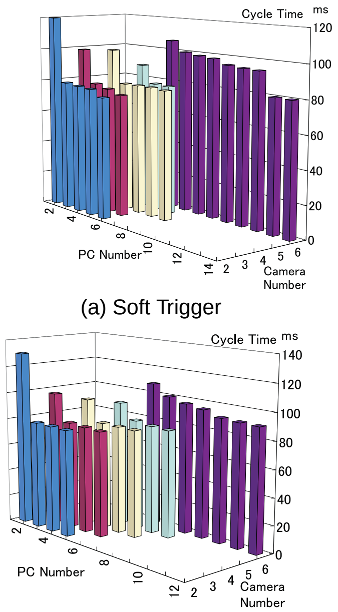


Figure 7. Average computation time for each pipeline stage.



(a) Soft Trigger

(b) Hard Trigger

Figure 8. Computation time for reconstructing one 3D shape.

In our PC cluster system, we developed two methods for synchronizing multi-view video capturing: use an external trigger generator (aka **Hard Trigger**) and control the cameras through network-communication (aka **Soft Trigger**). The performance was evaluated for both methods in Figure 8 (a) and (b).

From Figure 8 we can get the following observations: 1) In both synchronization methods, the rate saturates at a constant value in all cases. For Soft Trigger, the value is about 75 $\sim$ 80 ms, and for Hard Trigger about 80 $\sim$ 90ms. 2) Comparing the hard and soft triggers, the latter shows a slightly better performance.

Since as was proved in the first experiment, the throughput of the pipelined computation is about 30ms, the elongated overall throughput is due to the speed of *Image Capture* stage. That is, although a camera itself can capture images at a rate of 30 fps individually, the synchronization reduces its frame rate down into half. This is partly because the external trigger for synchronization is not synchronized with the internal hardware cycle of the camera and partly because it takes some time to transfer image data to PC memory.

3. Real-Time 3D Shape Reconstruction with Camera Actions

While not noted explicitly so far, the plane-based volume intersection method we proposed has a serious problem: when the optical axis of a camera is nearly parallel to the base plane, the size of the projected silhouette becomes very huge, which damages the computational efficiency. In the worst case, i.e. the optical axis becomes parallel to the base plane, the projection cannot be computed and the method breaks down.

In the case of static camera arrangements, it is possible to select one unique base plane to avoid the worst case. But the question of which base plane is optimal for computation, remains open. In the case of dynamic camera arrangements, we have to extend the method to avoid the worst case while doing the reconstruction efficiently.

3.1. The Computational Cost for Base Silhouette Generation

Since the computational cost for the base silhouette generation is ruled by the area size of the projected silhouette, we firstly calculate the area size as follows.

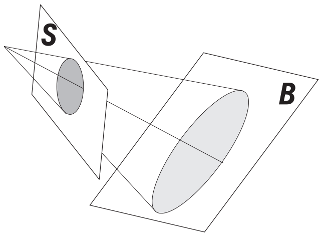


Figure 9. Projection from the Input Image Screen to a Plane

In Figure 9, an observed silhouette on the input image screen S is represented as a circle of radius r. This silhouette is projected onto a plane B as an ellipse. Let θ, f, l denote the dihedral angle between B and S, focal length, and the distance from the projection center to B respectively. The area size s of the projected silhouette is:

$$\begin{aligned} s &= \pi r^2 \cdot R(\theta, f, l) \qquad (1) \\ R(\theta, f, l) &= \frac{l^2}{f^2 \cos\theta} \end{aligned}$$

Thus, the projected silhouette is extended in the ratio of $R(\theta, f, l)$. Figure 10 shows the graph of R for $\theta \in [0, \pi/2)$. It is clear that R is monotonous increasing on θ and diverges to infinity at $\theta = \pi/2$, which corresponds to the worst case.

In the case of a static arrangement of n cameras, let $\{f_1, f_2, \ldots, f_n\}$, $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n\}$ and $\{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ denote the focal lengths, directions and the positions of the cameras respectively. A base plane can be represented by its normal vector $\mathbf{D} = \{D_x, D_y, D_z\}^t$ from the origin of the world coordinate system. Given a plane, for each camera,

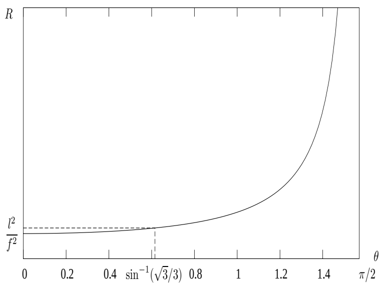


Figure 10. The Size Ratio between the Projected Silhouette and the Observed Silhouette

the angle θ_i and l_i can be calculated from $\mathbf{v}_i$, $\mathbf{p}_i$ and $\mathbf{D}$. Let $\{a_1, a_2, \ldots, a_n\}$ denote the area sizes of object silhouettes observed by the cameras. From equation 2, the area size of each projected silhouette becomes:

$$s_i = a_i \cdot R_i = a_i \cdot \frac{l_i^2}{f_i^2 \cos\theta_i} \qquad (2)$$

So the optimal selection of the base plane can be achieved by solving following optimization problem:

$$\mathbf{D} = \text{argmin} \sum_{i=1}^{n} s_i \qquad (3)$$

It is hard to solve the problem analytically.

In the case of dynamic camera arrangements, since the locations of cameras change dynamically, we can simply extend the notations as follows: let $\{\mathbf{v}_1(t), \mathbf{v}_2(t), \ldots, \mathbf{v}_n(t)\}$ and $\{\mathbf{p}_1(t), \mathbf{p}_2(t), \ldots, \mathbf{p}_n(t)\}$ denote directions and positions of the cameras at t respectively. Also the sizes of the observed silhouettes change and are denoted as $\{a_1(t), a_2(t), \ldots, a_n(t)\}$. Similarly, the base plane can be determined by solving the following optimization problem for each frame.

$$\begin{aligned} \mathbf{D}(t) &= \text{argmin} \sum_{i=1}^{n} s_i(t) \\ &= \text{argmin} \sum_{i=1}^{n} \left(a_i(t) \cdot \frac{l_i(t)^2}{f_i^2 \cos\theta_i(t)} \right), \qquad (4) \\ &\quad \text{where } \forall \theta_i(t) \neq \pi/2 \end{aligned}$$

Such simple extension is not efficient for a real-time 3D shape reconstruction because:

- For every frame, overheads are computational introduced to solve the optimization problem.
- As mentioned before, the volume is represented as cross-sections on parallel slices. So the coordinates of the cross-sections are represented in the same coordinate system as the base plane. From equation 4, the solution $\mathbf{D}$ changes temporally, i.e. volumes at different t are represented in different coordinate systems. This causes the overheads for coordinates conversion for every frame.

3.2. Limiting the Computational Cost by Dynamic Selection of the Base Plane

To realize the real-time 3D shape reconstruction with camera actions, we extend the plane-based volume intersection to a 3-base-plane method. The outline of the method is shown as follows:

1. Let $\mathbf{D}_1(t)$, $\mathbf{D}_2(t)$, $\mathbf{D}_3(t)$ denote the pre-defined 3 planes which are not parallel with each other.

2. For each camera, calculate the angles between the camera direction and the normal vector of each of $\mathbf{D}_1(t), \mathbf{D}_2(t), \mathbf{D}_3(t)$. Let $\{\{\theta_{1_{\mathbf{D}_1(t)}}, \theta_{1_{\mathbf{D}_2(t)}}, \theta_{1_{\mathbf{D}_3(t)}}\}$, $\{\theta_{2_{\mathbf{D}_1(t)}}, \theta_{2_{\mathbf{D}_2(t)}}, \theta_{2_{\mathbf{D}_3(t)}}\}$, ..., $\{\theta_{n_{\mathbf{D}_1(t)}}, \theta_{n_{\mathbf{D}_2(t)}}, \theta_{n_{\mathbf{D}_3(t)}}\}\}$ denote these angles.

3. Determine the base plane $\mathbf{B}_i(t)$ for each camera as:

   $$\mathbf{B}_i(t) = \begin{cases} \mathbf{D}_1(t), \text{if} & \theta_{i_{\mathbf{D}_1(t)}} = \min(\theta_{i_{\mathbf{D}_1(t)}}, \theta_{i_{\mathbf{D}_2(t)}}, \theta_{i_{\mathbf{D}_3(t)}}) \\ \mathbf{D}_2(t), \text{if} & \theta_{i_{\mathbf{D}_2(t)}} = \min(\theta_{i_{\mathbf{D}_1(t)}}, \theta_{i_{\mathbf{D}_2(t)}}, \theta_{i_{\mathbf{D}_3(t)}}) \\ \mathbf{D}_3(t), \text{if} & \theta_{i_{\mathbf{D}_3(t)}} = \min(\theta_{i_{\mathbf{D}_1(t)}}, \theta_{i_{\mathbf{D}_2(t)}}, \theta_{i_{\mathbf{D}_3(t)}}) \end{cases} \qquad (5)$$

   So we have cameras separated into 3 groups. Cameras in each group share the same base plane.

4. For each camera group, execute the plane-based volume intersection method and then 3 volumes are obtained.

5. Calculate the intersection of the 3 volumes which represents the object volume.

Since the 3 planes are not parallel with each other, it is clear that by this extension, the worst case can be avoided. The extension causes the following computational overheads:

- The computational cost for determining the base plane for each camera. However, the cost can be ignored for the operation is very simple and not pixel(voxel)-wise.
- The calculation for the intersection of the 3 volumes. As mentioned before, the result of the plane-based volume intersection is represented in the same coordinate system as the base plane. To get the intersection of the results on different base planes, coordinate conversion is required.

To decrease the computational cost of the coordinate conversion between the volumes on the 3 base planes, we can select 3 planes which are perpendicular to each other, i.e. $\mathbf{D}_1(t) \perp \mathbf{D}_2(t) \perp \mathbf{D}_3(t)$. So that, the coordinate conversion becomes simply switching among three coordinates.

Furthermore, when $\mathbf{D}_1(t) \perp \mathbf{D}_2(t) \perp \mathbf{D}_3(t)$, for each camera,

$$\min(\theta_{i_{\mathbf{D}_1(t)}}, \theta_{i_{\mathbf{D}_2(t)}}, \theta_{i_{\mathbf{D}_3(t)}})) \leq \sin^{-1}(\sqrt{3}/3) \qquad (6)$$

From the equation 2, the projected area size for each camera is limited to $\frac{\sqrt{6}}{2} \cdot \frac{l_i(t)^2}{f_i^2} a_i(t)$. The ratio in which the size of the observed silhouette is extended in this case is also plotted in Figure 10. In practice, since $l_i(t), f_i, a_i(t)$ can be regarded as bounded, the size of the projected silhouette for each camera is guaranteed to be limited. That is, by this extension, not only the worst case can be avoided, but also we can estimate the upper bound of the computational cost, which is very important for the design of the real-time active 3D shape reconstruction.

4. Conclusion

In this paper, we first described the parallelized volume intersection method, by which the real-time 3D object behavior reconstruction system is implemented on a PC cluster. The quantitative performance evaluations demonstrated that the acceleration and parallelizing algorithms we proposed are very efficient and real-time dynamic 3D shape reconstruction is realized.

Secondarily, we had discussion on how the computational cost changes when the camera direction changes, which is caused by the active object tracking. Based on the discussion, we extended the plane-based volume intersection method to a 3-base-plane method. The overheads of the extension is considerable low and the upper bound of the computational cost is clarified in the case of dynamic camera arrangements.

Currently we are developing active camera control methods to reconstruct dynamic object 3D shapes in high resolution.

This work was supported by the grant-in-aid for scientific research (A) 13308017. We are grateful to Real World Computing Partnership, Japan for allowing us to use their multi-viewpoint video data.

References

[1] T.Matsuyama, X.Wu, T.Takai, and S.Nobuhara: Real-Time Generation and High Fidelity Visualization of 3D Video: Proceedings of Mirage 2003, pp.1–10.

[2] T.Wada, X.Wu, S.Tokai, T.Matsuyama: Homography Based Parallel Volume Intersection: Toward Real-Time Reconstruction Using Active Camera: CAMP2000 Computer Architectures for Machine Perception, pp.331–339.

[3] E. Borovikov and L. Davis: A Distributed System for Real-Time Volume Reconstruction, Proc. of Computer Architectures for Machine Perception, pp.183-189, 2000.

[4] G.Cheung and T.Kanade: A Real Time System for Robust 3D Voxel Reconstruction of Human Motions, Proc. of CVPR, pp.714-720, 2000.

[5] T.Kanade, P.Rander, S.Vedula, and H.Saito: Virtualized Reality: Digitizing a 3D Time-Varying Event as is and in Real Time, in Mixed Reality (Y.Ohta and H.Tamura eds.), pp.41-57, Ohmsha, 1999.

[6] S.Moezzi, L.Tai, and P.Gerard: Virtual View Generation for 3D Digital Video, IEEE Multimedia, pp.18-26, 1997.

[7] T.Matsuyama and R.Yamashita: Requirements for Standardization of 3D Video, ISO/IEC JTC1/SC29/WG11, MPEG2002/M8107, 2002.

[8] Matsuyama, T.: "Cooperative Distributed Vision – Dynamic Integration of Visual Perception, Action, and Communication –," Proc. of Image Understanding Workshop, pp. 365-384, 1998

[9] H. Baker. Three-dimensional modelling. In *Fifth International Joint Conference on Artificial Intelligence*, pages 649–655, 1977.

[10] B. G. Baumgart. Geometric modeling for computer vision. Technical Report AIM-249, Artificial Intelligence Laboratory, Stanford University, October 1974.

[11] R. T. Collins. A space-sweep approach to true multi-image matching. In *IEEE Computer Vision and Pattern Recognition*, pages 358–363, 1996.

[12] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. In *IEEE International Conference on Computer Vision*, pages 307–314, 1999.

[13] A. Laurentini. How far 3d shapes can be understood from 2d silhouettes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):188–195, 1995.

[14] W. N. Martin and J. K. Aggarwal. Volumetric description of objects from multiple views. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(2):150–158, 1987.

[15] M. Potmesil. Generating octree models of 3d objects from their silhouettes in a sequence of images. *Computer Vision,Graphics, and Image Processing*, 40:1–29, 1987.

[16] J. Semple and G. Kneebone. *Algebraic Projective Geometry*. Oxford Science Publication, 1952.

[17] P. Srivasan, P. Liang, and S. Hackwood. Computational geometric methods in volumetric intersections for 3d reconstruction. *Pattern Recognition*, 23(8):843–857, 1990.

[18] R. Szeliski. Rapid octree construction from image sequences. *CVGIP: Image Understanding*, 58(1):23–32, 1993.