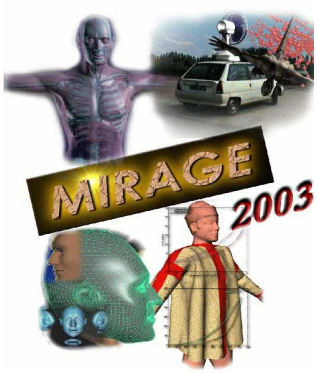
Mirage 2003 Proceedings

Computer Vision / Computer Graphics Collaboration for Model-based Imaging, Rendering, image Analysis and Graphical special Effects

INRIA Rocquencourt, France, March, 10-11 2003

http://telin.rug.ac.be/mirage2003



Jacques Blanc-Talon, André Gagalowicz, Philippe Gérard,
Wilfried Philips (editor), Dominique Potherat

Proceedings of Mirage 2003, INRIA Rocquencourt, France, March, 10-11 2003

MODELING AND RENDERING IN 3D COORDINATE DEFINED BY TWO CAMERAS FOR SHARED VIRTUAL SPACE COMMUNICATION

[1]*Daisuke Iso,* [1,2]*Hideo Saito*

[1]{iso, saito}@ozawa.ics.keio.ac.jp
[1]Department of Information and Computer Science, Keio University
3-14-1 Hiyoshi Kouhoku-ku Yokohama 223-8522, Japan
[2]PRESTO, Japan Science and Technology Corporation(JST)

ABSTRACT

In this paper, we describe an arbitrary view generation system using reconstructed voxel model in 3D space defined by two cameras. Saito and Kanade have proposed a method to define 3D space by using two camera-rays starting at the image's center. However, the 3D space was skew because it was generated based on projective geometry. In this paper, we propese a new method to reduce the skewness, so that the reconstructed model can be approximately represented in the Euclidean space without camera calibration. For making the reconstruction faster, we use Shape from Silhouette method with our new octree algorithm. In addition, we introduce an improved background subtraction algorithm by using not only color images but also disparity images to generate silhouette images with few noises. After the reconstruction, we color the model which consists of a set of voxels and render the model for synthesizing an arbitrary view image. Using this method, we build an application that composes two models reconstructed at two remote sites in a shared virtual space and being visualized by arbitrary view image.

1. INTRODUCTION

In 1999, many people in the world were surprised at a movie. "The matrix" [1] presented some scenes that view point moved freely in the world that time passed very slowly. Unfortunately, these amazing scenes called "Bullet Time Walk Through" were constructed by hand-made CG effects that cost much time and human work. For such backgrounds, the techniques for automatic generation of arbitrary view image from multiple real images are recently developed.

In the past decade, efforts have been put in developing methods for reconstructing 3D models of real world from multiple cameras. Shape from Silhouette[2] is a method using object silhouettes extracted from camera images and reconstructs the objects to separate 3D space based on information of silhouettes and cameras.

Representation of 3D space or shape is also discussed. Baker[3] built a wire-frame model from multiple silhouette images. One of the benefits of wire-frame model is its simplicity. Only vertices of triangles are required for construction. Martin and Aggarwal[4] represented the model as 3D volume data. It can represent accurate 3D models, but it took a long time to process all voxels. Potmesil[5] proposed octree. The octree structure is an 8-ary tree structure generated by a recursive subdivision of the whole 3D space called as "*universal cube*" into octants until homogeneous blocks of voxels are reached. The feature of octree structure is less computation.

Using these 3D shape representations, 3D shape reconstructing systems are actively researched. Kanade[6] proposed the method of generating arbitrary view of time varying events with modeling from multiple cameras. However, this method can't run in real-time because the processes are complex. Cheung[7] et al. presented a multi-PC/camera system that can perform 3D reconstruction and ellipsoids fitting of moving humans in real time. However, they focused on the human motion tracking, so they did not render virtual view images in real time. Image Based Visual Hulls[8] was proposed by Matusik et al. in recent years. They realized real-time arbitrary view image generation system using their own Visual-Hull computation algorithm. However, their system needs camera calibration.

We propose the system automatically reconstructs real world objects on computer using multiple view camera images. It runs about 5 frames per seconds in our current system configuration. Using two cameras for defining world coordinates, our system doesn't need camera calibration. The result of the reconstruction is displayed as an arbitrary view image.

An overview of our system and algorithm is given in Section 2, followed by detailed description of each process in Section 3 and 4. The result of experiments is in Section 5 and an application using our method is described in Section 6. Finally, conclusion and future work are provided in Section 7.

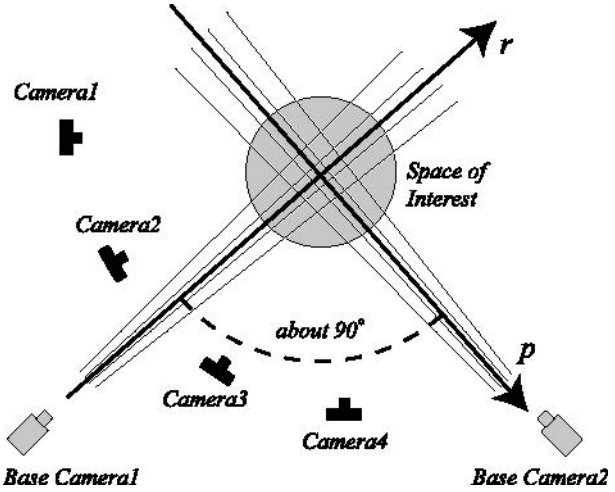


Figure 1: Two base cameras and PGS coordinates

2. OVERALL SYSTEM AND ALGORITHM

Our system consists of four stereo cameras and five PCs connected to the LAN. When we build world coordinates, we use other two cameras. Four PCs for capturing are Pentium 3 processor 750MHz. A PC for generating arbitrary view image is Dual-Xeon processor 1.8MHz. The LAN is Gigabit-Ethernet. The stereo camera called "Color Digiclops"[9] by Point Grey Research Corp. consists of three CCD cameras and can capture color image and disparity image at the same time. It creates a disparity image with the combination of the two-baseline stereo method. Each camera is connected a PC which locally extracts the silhouettes from the image captured by the camera.

We use not only color images but also disparity images to generate silhouette. As the result of this, we can perform robust background subtraction. We don't use these disparity images for model reconstruction because disparity values in each image are too rough to reconstruct model. After silhouette image generation, JPEG compression is carried out to the silhouette images to reduce image data size and camera images and they are then sent to host computer to perform 3D reconstruction. If these images are not compressed, high-traffic situation is caused because host computer receives many color and silhouette images per second from capturing PCs. Just to avoid such situation, the images are converted into compressed JPEG format.

We use octree generation algorithm[5, 10] with Shape from Silhouette method to reconstruct 3D object shape because of its runtime, and improved it. After generating octree, we convert octree to voxel models and perform internal voxel removal, because we need only surface to display result model. After coloring of voxels, we render surface voxel model as an arbitrary view image.

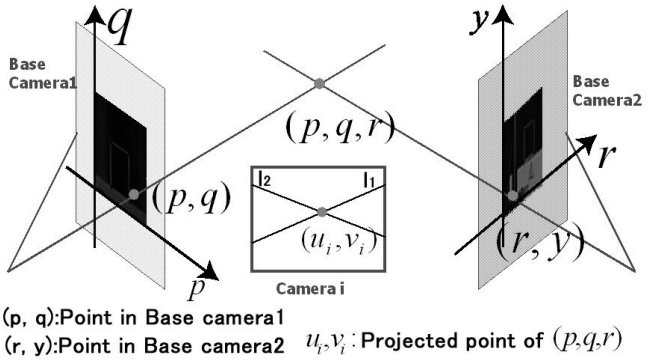


Figure 2: Projective Grid Space

Our system doesn't need camera calibration because it uses the epipolar lines by which other two camera-rays are projected on four cameras by Fundamental Matrix as coordinate axes. In Section 3, we describe about generating 3D space defined by two cameras in detail.

3. DEFINITION OF 3D SPACE

Our system defines 3D world space without camera calibration and reconstructs 3D model in this space. Now, we describe about definition of 3D space in this section.

In common, a set of (at least 6) known 3D points in world coordinates and their corresponding projections in camera images are required to compute camera parameters for 3D-2D correspondences. Especially in huge space, camera calibration is very hard work because many 3D points in the space need to be surveyed.

Saito and Kanade[11] proposed 3D model reconstruction in "Projective Grid Space(PGS)" generated by only two base cameras. However, because the coordinates are too skew to render the model correctly, the information of model shape is only used for Image-based rendering.

For reducing the skew effect in PGS, we propose to employ other two cameras with almost infinite, so that we can build less skew Projective Grid Space using these camera-rays as coordinate axes.

In our method, all we need to do is surveying only 2D points in each image plane because we need only epipolar lines, so the cost to survey is the same even in large-scale space. In addition, this space can't be skew because the coordinate axes are built with the rays of the base cameras which projection can be assumed orthogonal.

Figure 1 shows the environment seen from the top. We call the two cameras which define PGS as "base camera". As shown in the figure, these two base cameras are placed in the distance to capture the space of interest as big as they can.

We name each base camera to BC1, BC2. A point $\mathbf{P}(p, q, r)$

in PSG is defined by two points in image coordinates, one is $\mathbf{p}_1(p,q)$ in BC1 and another is $\mathbf{p}_2(r,y)$ on the epipolar line which is the projection of the camera ray through $\mathbf{p}_1$. Here, y is calculated by linear equation of the epipolar line. We calculate the epipolar line with fundamental matrix computed by Zhang's method[12].
In our method, we also compute other 3D-2D correspondences between PGS to images captured by the stereo cameras. The ray through the point $\mathbf{p}_1(p,q)$ in BC1 image is projected as the epipolar line $\mathbf{l}_1$ in ith camera by following equation,

$$\mathbf{l}_1 = \mathbf{F}_{1i} \begin{bmatrix} p \\ q \\ 1 \end{bmatrix} \tag{1}$$

where, $\mathbf{F}_{1i}$ is the fundamental matrix from BC1 to ith camera. In the same way, the ray through the point $\mathbf{p}_2(r,y)$ in BC2 image is projected as the epipolar line $\mathbf{l}_2$ in ith camera by the fundamental matrix $\mathbf{F}_{2i}$,

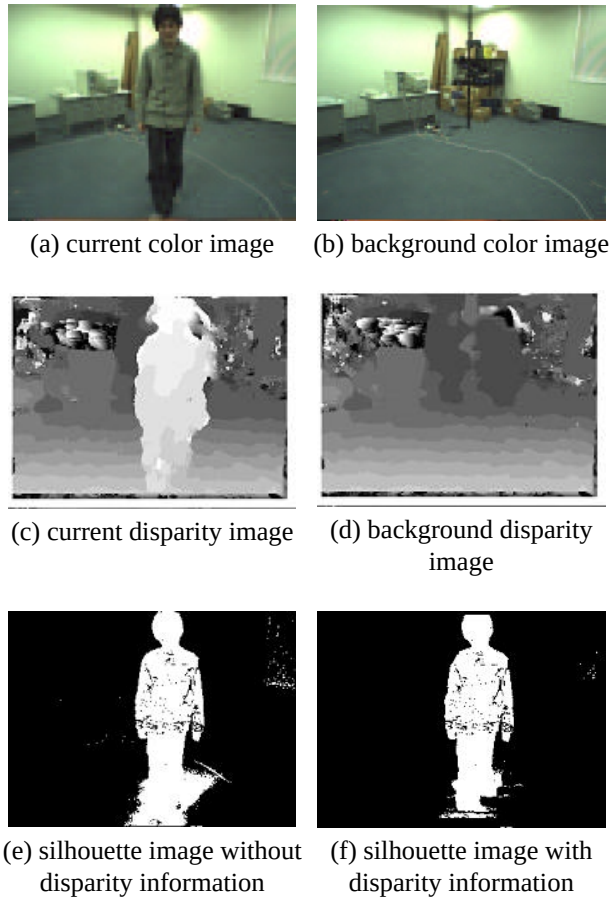
$$\mathbf{l}_2 = \mathbf{F}_{2i} \begin{bmatrix} r \\ y \\ 1 \end{bmatrix} \tag{2}$$

where, the notation $\mathbf{F}_{2i}$ represents the fundamental matrix between BC2 and ith camera. By calculating the intersection of $\mathbf{l}_1$ and $\mathbf{l}_2$, we compute a correspondence between $\mathbf{P}$ and a point in ith camera. Figure 2 shows the intersection of two epipolar lines.
In this way, projected 2D position on arbitrary camera can be computed for every 3D point in the PGS. Since such projecting computation is repeated many times for every 3D point in 3D shape reconstruction, we keep the 3D-2D projection relationship for every 3D position in a lookup table for reducing the computation time. Because two discrete points in the base camera images define a point in PGS, and the reconstructed 3D volume is limited by the FOV of every camera, the entry number of the lookup table can be determined as a certain finite number, which does not require huge memory space. For example, the size of the lookup table is about 268 MByte in our experimental setup.

4. 3D SHAPE RECONSTRUCTION

4.1. Silhouette image generation

Before 3D shape reconstruction, we must generate silhouette images by background subtraction. Here, we describe our method of silhouette image generation.
We use both color and disparity information taken by stereo camera for silhouette image generation. Only color information, speculars and shadows affect result images. On the other hand, disparity images are insensitive for these effects, but we can get only coarse silhouette. So we perform background subtraction by combining these two information.



(a) current color image (b) background color image

(c) current disparity image (d) background disparity image

(e) silhouette image without disparity information (f) silhouette image with disparity information

Figure 3: silhouette generation

Given a background image whose (x,y) pixel color is denoted by $\mathbf{c_b}(x,y)$, and disparity value is $d_b(x,y)$, and current image pixel color is $\mathbf{c_c}(x,y)$ and disparity value $d_c(x,y)$, we perform silhouette generation as follows:

if $|d_b(x,y) - d_c(x,y)| > th_D$ **then**
  if $\|\mathbf{c_b}(x,y) - \mathbf{c_c}(x,y)\| > th_U$ **then**
    $p(x,y) = SILHOUETTE$
  else
    if $\|\mathbf{c_b}(x,y) - \mathbf{c_c}(x,y)\| < th_L$ **then**
      $p(x,y) = NOT\_SILHOUETTE$
    else
      $\theta = \cos^{-1}\left(\frac{\mathbf{c_b}(x,y)\cdot\mathbf{c_c}(x,y)}{\|\mathbf{c_b}(x,y)\|\|\mathbf{c_c}(x,y)\|}\right)$
      if $\theta > th_A$ **then**
        $p(x,y) = SILHOUETTE$
      else
        $p(x,y) = NOT\_SILHOUETTE$
else
  $p(x,y) = NOT\_SILHOUETTE$

where, $p(x, y)$ is the pixel condition of a silhouette image, and th_D, th_U, th_L and th_A are the disparity, upper(color), lower(color), and angle thresholds empirically defined.
The result of silhouette generation is shown in Figure 3. Compare Figure 3(f) to Fig 3(e), shadow effect around the foot in Figure 3(e) is removed in Figure 3(f) because disparity image is not affected by the shadow.

4.2. Shape from Silhouette

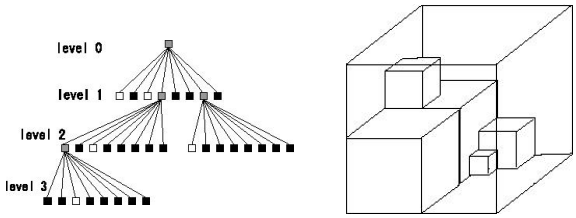
With correspondences between PGS to all camera images, we perform shape from silhouette using octree data structure, to reconstruct 3D object shape. In Euclidean space, perspective projection of a silhouette image generates a conic volume model. Resultant 3D model is generated by intersection of all conic 3D models. The equation of shape from silhouette is shown Equation 3.

$$V_I = \bigcap_{i \in I} V_i \tag{3}$$

where, I is denoted as a set of all silhouette images, and i is a image in the set. V_i is the object model generated from the ith image.
In general, each voxel in a Euclidean space is tested if it belongs to the inside of the silhouette by projecting itself onto all silhouette images. If a voxel is out of silhouette in an image, the voxel is not a part of model. In contrast, a voxel inside of silhouette in all silhouette images is a part of the model.
For applying the Shape from Silhouette method in PGS, we transform 3D point in PGS to 2D point in image coordinates using lookup table prepared in advance instead of perspective transform (in Section 3).

4.3. Octree generation

The octree structure is an 8-ary tree structure generated by a recursive subdivision of the whole 3D space called as "*universal cube*" into octants until homogeneous blocks of voxels are reached. It is shown in Figure 4.
If an octant does not entirely consist of the same type of voxels, then it is further subdivided until homogeneous cubes, possibly single voxels, are obtained. Allocating these cubes, and transforming them into all silhouette images, we perform the intersections of transformed cube region with silhouette regions. Then, we describe the detail of the method as follows.
At first, 8 vertices of "*universal cube*" are transformed into all image planes, and search region is decided. Then, transformed cube constitutes hexagon. However we intend to have fast reconstruction, so we search the minimum rectangle including a hexagon. After decision of the rectangle, we perform intersection check. Then, we don't

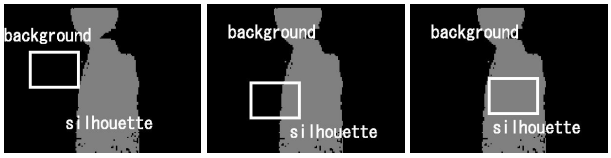


(a) Octree structure (b) Recursive cube subdivision

Figure 4: Octree structure and universal space it shows

search all pixels in the rectangle to save execution time. We show the intersection check in Figure 5.
If the region includes part of silhouette and background, the cube corresponding to the region is temporarily determined as "*GRAY*" which means partly object is included. And if all the pixels of the search region are inside the silhouette, the cube is temporarily decided as "*WHITE*". At the adverse case (all pixels in the region are background), the type is temporarily determined as "*BLACK*". Once the temporary type is determined as "*BLACK*", the conclusive cube type is decided as "*BLACK*" according to the concept of shape from silhouette. In other cases, intersection check of the cube is continued until all images are referred. After all images are referred, the conclusive cube type is determined. If all temporary cube types are "*WHITE*", the cube is eventually determined as "*WHITE*". The other case is determined as "*GRAY*". In this case, all temporary types are stored, and referred to child node to shorten computation time.
For example, if the temporary parent node types are determined as "*WHITE*","*WHITE*","*GRAY*","*GRAY*", the final cube type is determined as "*GRAY*". In this case, the cube is subdivided and has children, and the first and second temporary nodes are certainly determined as "*WHITE*". It shows in Figure 6. To refer stored types, we eliminate these losses. After the type of the cube is determined, process flow is branched depending on the cube type. If the type is "*BLACK*" or "*WHITE*", we don't subdivide the cube. It means that the octree node becomes leaf node. If the type is "*GRAY*", the cube is subdivided to 8 cubes. In short, the octree node newly makes children octree nodes. After determining the type of current cube, we recursively iterate the same process to the others. The pseud code is given as follows.

```
function makeOctree {
  for(i=1;i ≤ cameraNumber;i++){
    refer the parent temporary cube type;
    if referred type is ''GRAY" then
     project 8 vertices into image;
     decide search rectangle;
     search the rectangle;
     decide the temporary cube type;
```



(a) the case of *"BLACK"* (b) the case of *"GRAY"* (c) the case of *"WHITE"*

Figure 5: three types of intersection check. The white rectangle is search area.

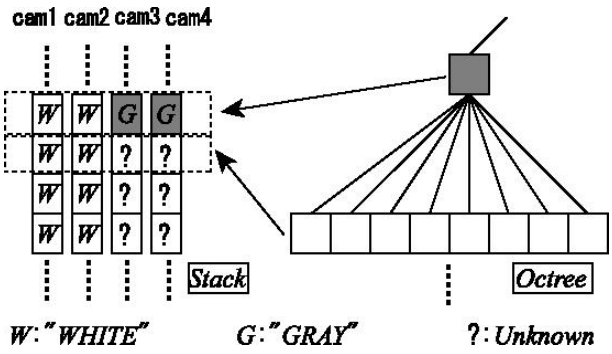


Figure 6: Reference of temporary parent type

```
    if temporary cube type is ''BLACK" then
      store the temporary type as ''BLACK";
      exit this loop;
    else if current cube type is not ''BLACK"
  then
      store the temporary type;
    else
     decide the temporary cube type as ''WHITE";
  }
  refer the all temporary cube types;
  if one of the temporary cube types is ''BLACK"
  then
    decide the node type as ''BLACK";
  else if all temporary cube types are ''WHITE"
  then
    decide the node type as ''WHITE";
  else
    decide the node type as ''GRAY";
    make 8 children node;
   for (j=1;j ≤ 8;j++) {
     makeOctree;
    }
}
```

4.4. Internal voxels removal

After an octree generation, we convert an octree model to voxel models for display. Each octree node has 2^{3n} voxels, where, n is the octree level. If we display all these voxels, rendering runs in much time. To avoid this problem, we remove internal voxels by the concept of 6-connectedness and display only surface model. Figure 7 shows about it.

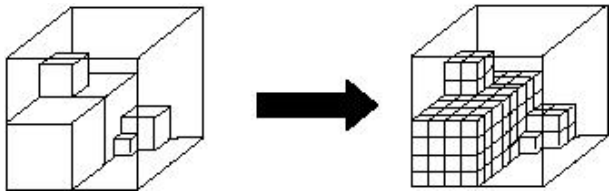


Figure 7: Internal voxel removal

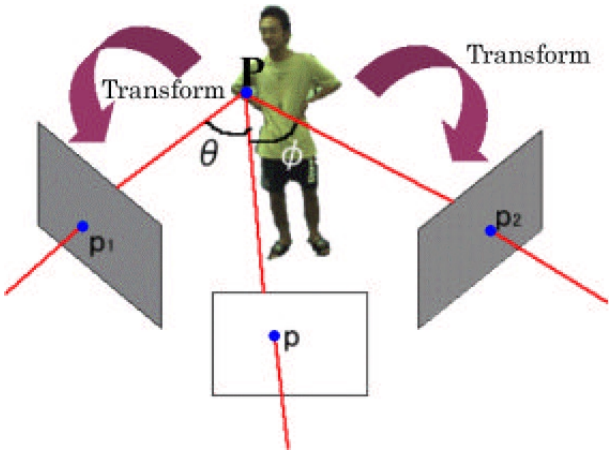


Figure 8: Coloring of voxels

A voxel $\mathbf{v} \in INNER$ is decided if one or more of its 6 connected neighbors are vacant voxels. Otherwise as $\mathbf{v} \in SURFACE$. Note, we denote the sets of all internal voxels as $INNER$, and sets of all surface voxels as $SURFACE$.

In the next step described in Section 4.5, we use only surface voxels.

4.5. Coloring of voxels

Generated 3D surface voxel model is not colored. In this step, voxel color is assigned. In our method, we dynamically blend the voxel color from two pixels of real camera images selected depending on virtual view position. It shows in Figure 8 and Equation 4.

$$c(\mathbf{p}) = \frac{\phi}{\theta+\phi} c_1(\mathbf{p}_1) + \frac{\theta}{\theta+\phi} c_2(\mathbf{p}_2) \qquad (4)$$

Where, θ and ϕ are horizontal angle between ith camera and virtual view, $i+1$th camera and virtual view respectively. The blending weight is determined by these angles. When a 3D point $\mathbf{P}$ is determined, $\mathbf{p}_1$ and $\mathbf{p}_2$ are transformed by the lookup table discribed in Section 3. Given two pixels $\mathbf{p}_1$ and $\mathbf{p}_2$ from correspondences, a virtual view

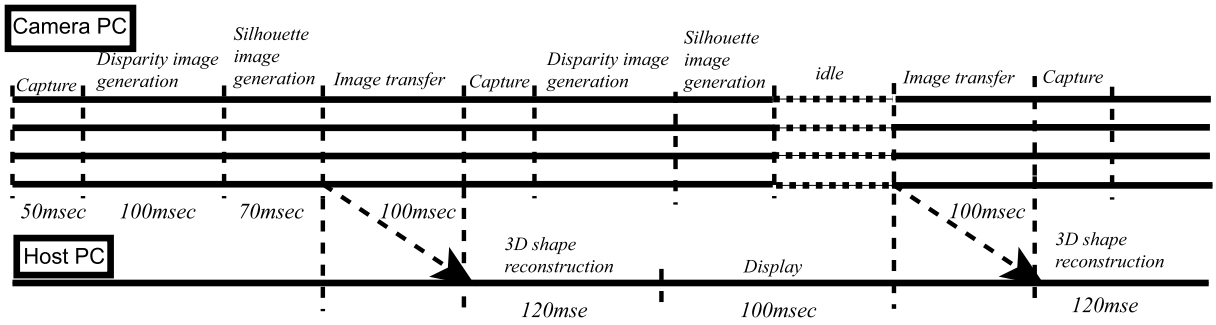


Figure 9: The timing diagram for pipeline processing

(a) camera 1 (b) camera 2 (c) camera 3 (d) camera 4

Figure 10: Multiple view images for generating arbitrary view image

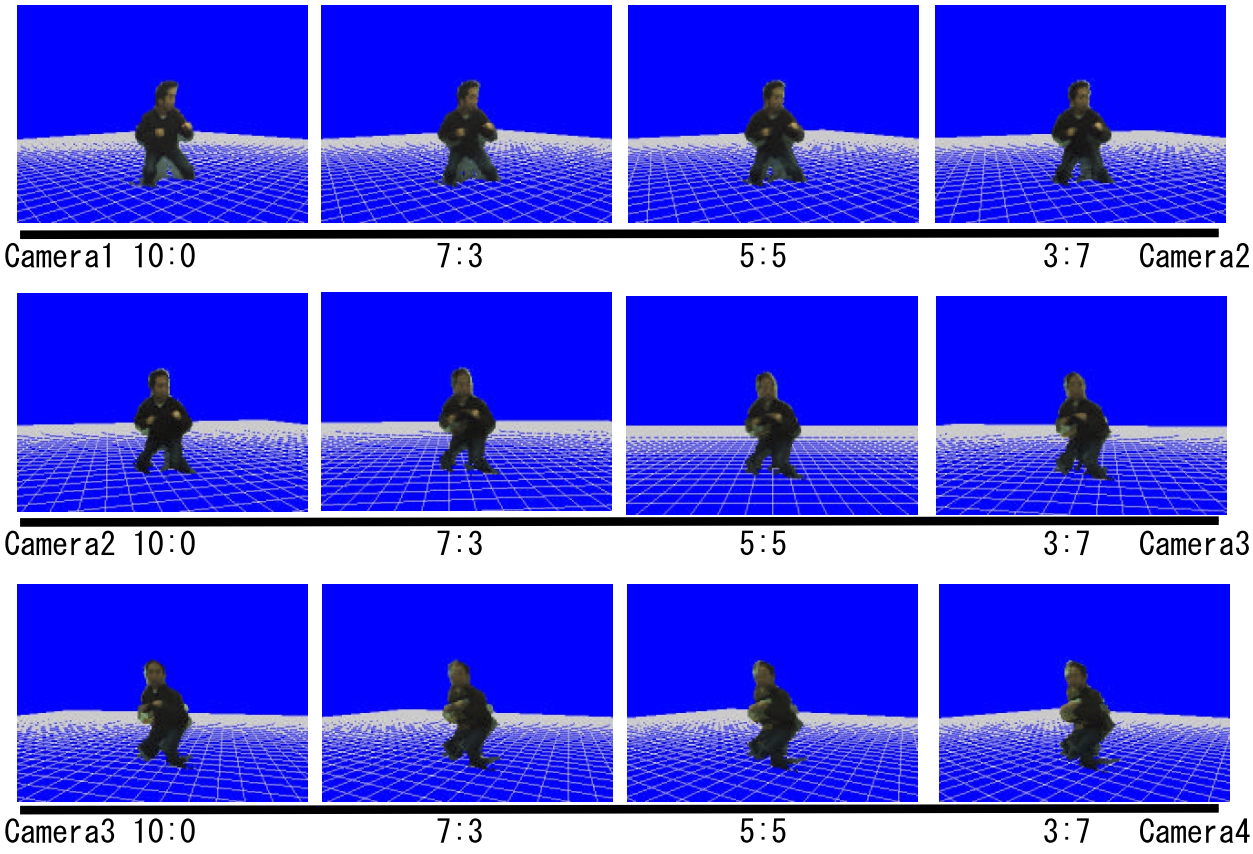
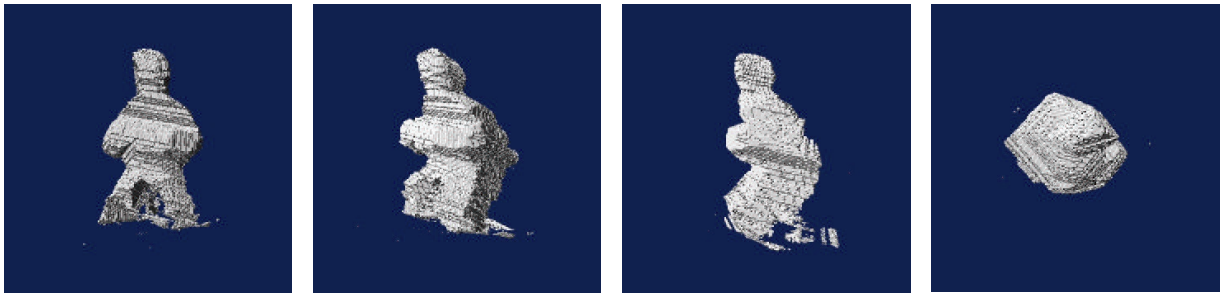
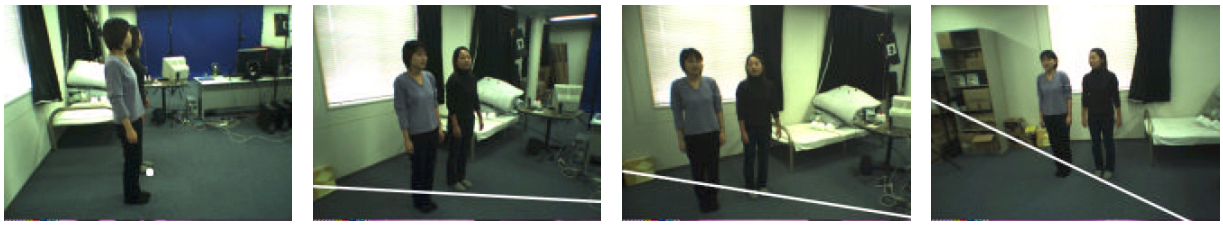


Figure 11: arbitrary view images



(a) view 1 (b) view 2 (c) view 3 (d) top view

Figure 12: Reconstructed shape represented by surface polygon model that is converted from voxel model using MC method

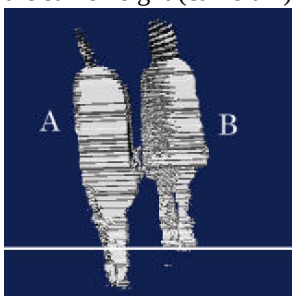


(a) Two people of almost the same height (camera 1)

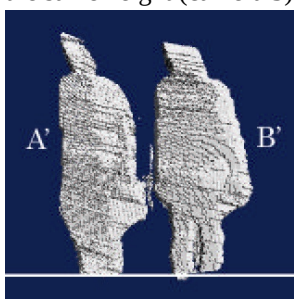
(b) Two people of almost the same height (camera 2)

(c) Two people of almost the same height (camera 3)

(d) Two people of almost the same height (camera 4)



(e) Reconstructed models by previous method



(f) Reconstructed models by our method

Figure 13: Difference of skewness

pixel color $c(\mathbf{p})$ is calculated by these colors $c(\mathbf{p}_1)$ and $c(\mathbf{p}_2)$ according to Equation 4. After coloring of voxels, we render the model as arbitrary view images. With a virtual point of view, we compute the projection matrix and render the resulting model. To render faster, we use "Microsoft DirectX"[13], which carry out most of the rendering computations on the hardware, such as depth-buffer composite, projection transform and so on.

5. EXPERIMENT

5.1. Settings

We experiment our system with the following settings:

- the number of voxels: 256 x 256 x 256
- maximum octree level: 8
- image resolution: 320 x 240 pixels
- color depth: 24bit
- disparity depth: 8bit

Note that, our method cannot know the size of voxel in real world because we build virtual space using only correspondences in images.

5.2. Result

The timing diagram for the pipeline processing is given in Figure 9. "Display" in Figure 9 includes the time of convert from the octree model to the voxel model and the coloring of voxels. "Image transfer" is the time of JPEG compression and the transfer of color images and silhouette images from the capture PC to the host PC. The time of "3D shape reconstruction" is 120msec because we use improved octree generation algorithm, i.e., our implementation of Shape from Silhouette not using octree algorithm takes 5200msec.
Figure 10 shows multiple real images, and the virtual views are shown in Figure 11.The ratio under the line is the blending weights between two pixels. The reconstructed shape is shown in Figure 12. In this figure, the shape is represented by surface polygon model that is converted from the voxel model using MC method[14] to see it clearly. In spite of the reconstruction in Projective Grid Space, our method can reconstruct the model which looks like in Euclidean space model.
The comparison between Figure 13(e) and (f) demonstrates the reduction of the skew effect of the reconstruction in the proposed Projective Grid Space defined by the extra base cameras. The shape shown in Figure 13(e) is reconstructed from the four images shown in Figure (a),(b),(c), (d), in the PGS defined by the camera 1 and camera 4. On the other hand, the extra base cameras are introduced for definition of the PGS, in which the shape shown in Figure 13(f) is reconstructed.
Because of the skew effect by the camera 1 and camera 4, the shape shown in Figure 13(e) is distorted. Although the two people are almost same height, the reconstructed size of each girl is different. The height of the model A is 178 voxels while the height of the model B is 158 voxels.
Contrary to this, since the epipolar lines from the extra base cameras are almost parallel, the shapes of the two people have almost the same size in the reconstructed model shown in Figure 13(f). The height of the model A' is about 209 voxels and the height of the model B' is 201 voxels.
The line in Figure 13(e) represents a section of the p-r plane in PGS. One of the lines on the p-r plane is shown as the epipolar lines in Figure 13(b), (c), (d) and a point in Figure 13(a). The model A stands on the p-r plane while the foot of the model B is under the plane. On the other hand, a section of the p-r plane in our PGS defined by the extra base cameras is indicated as the line in Figure 13(f). Both A' and B' are standing on the same p-r plane in our PGS.
The reason why the heads of the models in Figure 13(f) are skew is not as same as PGS skewness. It causes the errors of intersections of two epipolar lines. We survey 2D points in base camera image. They capture narrow area because the focuses of the cameras are infinite, so when the point is far from the center of base camera image, the epipolar line is misaligned from correct position. In fact, 3D-2D correspondences are incorrect.

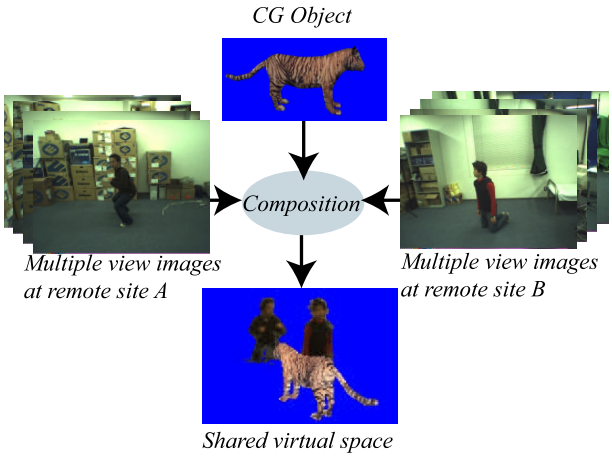


Figure 14: A communication application in shared virtual space

6. APPLICATION

Using our PGS generation method, we build a communication application in shared virtual space. Figure 14

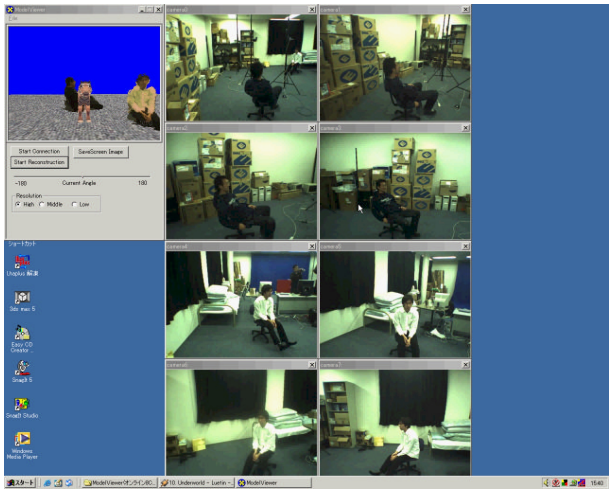


Figure 15: A screen capture of our system which is running

shows the concept.
This application captures multiple view image sequences at remote two sites and composes two reconstructed models using the images and CG object prepared in advance. Capture PCs capture the targets at each site and generate silhouette images using color image and disparity image. Color and silhouette image sequences are then sent to the host computer through Gigabit Ethernet and used for model reconstruction. At host computer, the model is reconstructed by the set of four silhouette images captured at each remote point and colored depending on virtual viewpoint. In addition, CG object, which is polygon model with textures, is prepared. Figure 15 shows the screen capture of the system which is running. The window drawn at upper left in the Figure 15 shows the shared virtual space. Other windows in the Figure 15 show the current camera images.
In the future, when baud rate will be improved, people at any points can play each other anytime and anywhere.

7. CONCLUSION AND FUTURE WORK

We proposed a fast 3D shape reconstructing system from multiple view images using Octree and Shape from Silhouette without camera calibration, and show the result image. Our system consists of 4 cameras connected to four PCs individually and a host PC for reconstruction. Each camera captures color and disparity images and generate silhouette image using both color and disparity image. Host PC collects these silhouette images sent from local PCs and reconstruct 3D shape with octree data structure. Reconstructed model is converted to voxel surface model and displayed in arbitrary view. Two base cameras build non-skew Projective Grid Space. Moreover, these two cameras enable none camera calibration.
The problem of our system is that the region of few errors of intersections is limited because the FOV (Field Of View) of two base cameras is a part of the FOV of cameras for model reconstruction. The more 3D point in PGS separates from the center of base camera's FOV, the more epipolar lines, which becomes PGS coordinates, shifts from correct position because 2D points in base camera images used for computing fundamental matrices are surveyed in a narrow range. When the error of epipolar line becomes large, the errors of intersections between two epipolar lines become large. To avoid this problem, we set two base cameras far from space of interest and fit it in field angle of the cameras.
Total time of generating an arbitrary view image is about 5 frames per seconds in our current system configuration. However, real time arbitrary view generation systems exist in the world, for example IBHV[8]. By lowering voxel resolution, we can shorten reconstruction time in exchange for quality of arbitrary view image. However, it is important to shorten run-time and improve result image quality together. It is very difficult problems but we try to solve them.

8. REFERENCES

[1] http://www.whatisthematrix.com/, "The matrix", 1999.

[2] A. Laurentini, "How many 2D silhouetts does it takes to reconstruct a 3D object ?", *Computer Vision and Image Understanding*, vol. 67, pp. 81–87, 1997.

[3] H. Baker, "Three-dimensional modelling", 1977, pp. 649–655.

[4] W. N. Martin and J. K. Aggarwal, "Volumetric descriptions of objects from multiple views", *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-5*, vol. 2, pp. 150–158, 1983.

[5] M. Potmesil, "Generating octree models of 3D objects from their silhouettes in a sequence of images", *Computer Vision, Graphics, and Image Processing*, vol. 40, pp. 1–29, 1987.

[6] T. Kanade, P. W. Rander, S. Vedula, and H. Saito, "Virtualized Reality: Digitizing a 3D time-varying event as is and in real time", *International Symposium on Mixed Reality(ISMR99)*, pp. 41–57, 1999.

[7] G. K. M. Cheung, T. Kanade, J. Y. Bouguet, and M. Holler, "A real time system for robust 3D voxel reconstruction of human motions", *CVPR 2000.*

IEEE Comput. Soc, Los Alamitos, CA, USA, vol. 2, pp. 714–729, 2000.

[8] W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan, "Image Based Visual Hulls", in *SIGGRAPH 2000*, 2001.

[9] http://www.ptgrey.com/products/digiclops/index.htm, "Color digiclops", .

[10] R. Szeliski, "Rapid octree construction from image sequences", *CVGIP: Image Understanding*, vol. 58, pp. 23–32, 1993.

[11] H. Saito and T. Kanade, "Shape reconstruction in Projective Grid Space from a large number of images", *Proc. CVPR*, 1999.

[12] Z. Zhang, "Determining the epipolar geometry and its uncertainty:a review", Research Report 2927, INRIA Sophia-Antipolis, France, July 1996.

[13] http://www.microsoft.com/directx/, "Microsoft DirectX", .

[14] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm", in *SIGGRAPH 1987*, 1987, vol. 21, pp. 163–169.