3D shape reconstructing system from multiple-view images using octree and silhouette

Iso, Daisuke, Saito, Hideo, Ozawa, Shinji

Daisuke Iso, Hideo Saito, Shinji Ozawa, "3D shape reconstructing system from multiple-view images using octree and silhouette," Proc. SPIE 4572, Intelligent Robots and Computer Vision XX: Algorithms, Techniques, and Active Vision, (5 October 2001); doi: 10.1117/12.444174

Event: Intelligent Systems and Advanced Manufacturing, 2001, Boston, MA, United States

3D shape reconstructing system from multiple view images using octree and silhouette

Daisuke Iso and Hideo Saito† and Shinji Ozawa

Department of Information and Computer Science, Keio University
3-14-1 Hiyoshi Kouhoku-ku Yokohama 223-8522, Japan

ABSTRACT

In this paper, we describe the 3D shape reconstructing system from multiple view images using octree and silhouette. Our system consists of four calibrated cameras. Each camera is connected to a PC that locally extracts the silhouettes from the image captured by the camera. The four silhouette images and camera images are then sent to host computer to perform 3D reconstruction. For making the reconstruction faster, the object 3D space is represented by octree structure. 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 projecting them into all silhouette images, we perform the intersection of the projected cube region with silhouette region. We develop a new algorithm for fast speed constructing octree. The algorithm can reduce time complexity to check if a node should project 8 cube vertices to image plane, using a stack that keeps parents' temporary cube type. By using our algorithm, our system runs in semi real time computation, (about 5 frames per second) for generating 3D shape of the human in voxel representation.

Keywords: 3D Reconstruction, Multiple View Images, Shape from Silhouette, Camera Calibration, Octree

1. INTRODUCTION

Recent years, the system reconstructs real world on computers are needed in multimedia fields. For example, we can use for virtual museums and cyber shopping moles, games, etc. Most of them are constructed by handmade CG which costs much time and human work under the present conditions.

In the past decade, efforts have been put in developing methods for reconstructing 3D models of real world from multiple cameras. Shape from Silhouette[1] is the 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[2] built a wire-frame model from multiple silhouette images. One of the benefits of wire-frame model is less information. Only known nodes of triangles recover models by combining them. Martin and Aggarwal[3] represents the model as 3D volume data. It can represent accurate 3D models, but it takes a long time to operate all voxels. Potmesil[4] 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 this, octree structure costs less runtime.

Using these method, 3D shape reconstructing systems are actively researched. Kanade[5] 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 of process complexity. Real-time arbitrary view generating system consists of PC clusters is proposed by Wada.[6] However, it is very complecated system consists of a number of 16 PCs. German[7] 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.

We propose the system automatically reconstructs real world objects on computer in semi real-time using multiple view camera. The result of reconstruction is displayed as a 3D model with colors.

Email addresses:
Daisuke Iso: iso@ozawa.ics.keio.ac.jp
Hideo Saito: saito@ozawa.ics.keio.ac.jp
Shinji Ozawa: ozawa@ozawa.ics.keio.ac.jp
†PRESTO, Japan Science and Technology Corporation(JST)

Intelligent Robots and Computer Vision XX: Algorithms, Techniques, and Active Vision,
David P. Casasent, Ernest L. Hall, Editors, Proceedings of SPIE Vol. 4572 (2001)

A brief overview of our system and algorithm is given in Section 2, followed by detailed description of each process in Section 3, 4. The system architecture is discussed in Section 5. The result of experiments is in Section 6. Finally, conclusion and future work is in Section 7.

2. OVERALL SYSTEM

Our system consists of four calibrated cameras. Each camera is connected to a PC that locally extracts the silhouettes from the image captured by the camera. Not only color images, but also disparity images[8] we use to generate silhouette. As the result of this, we can perform robust background subtraction which has a head for color noise. The silhouette images and camera images are then sent to host computer to perform 3D reconstruction. We use octree generation algorithm[4,9] to reconstruct 3D object shape because of its runtime, and improved it. Furthermore, we parallelized the algorithm to generate each octant independently. After generating octree, we convert octree to voxel models and perform internal voxel removal, because we need only surface to display result model. Finally, we render surface voxel model.

3. CAMERA CALIBRATION

The perspective projection of the camera maps the 3D homogeneous world coordinate system $\mathbf{P}(X, Y, Z)$ into a 2D image coordinate $\mathbf{p}(x, y)$ by a transformation matrix. The transformation is computed by a Camera Calibration technique from a set of (at least 6) known 3D points and their corresponding image projections by solving system of 11 simultaneous linear equations(because of $p_{34} = 1$). It is given as

$$\begin{bmatrix} h \cdot x \\ h \cdot y \\ h \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{1}$$

where, h is scale factor, and $p_{11} \cdots p_{34}$ are camera parameters. Then, we use robot arm to find $\mathbf{P}$ and $\mathbf{p}$ as shown Fig 1. In our experiments, we use about over 450 corresponding points. Using these corresponding points, we solve camera parameters by least square method.

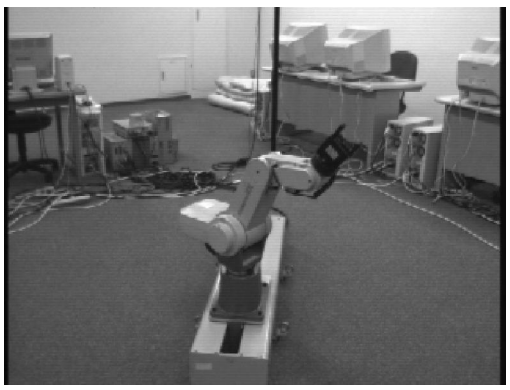


Figure 1. robot arm for camera calibration

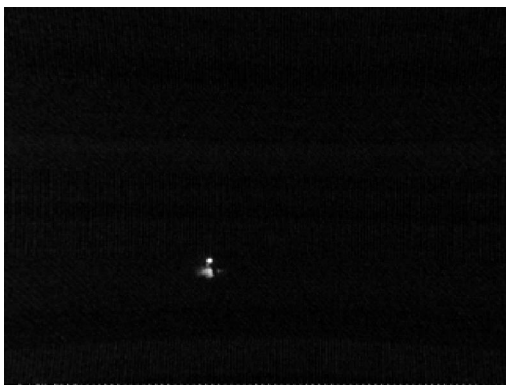


Figure 2. image for camera calibration

4. 3D SHAPE RECONSTRUCTION

4.1. Silhouette generation

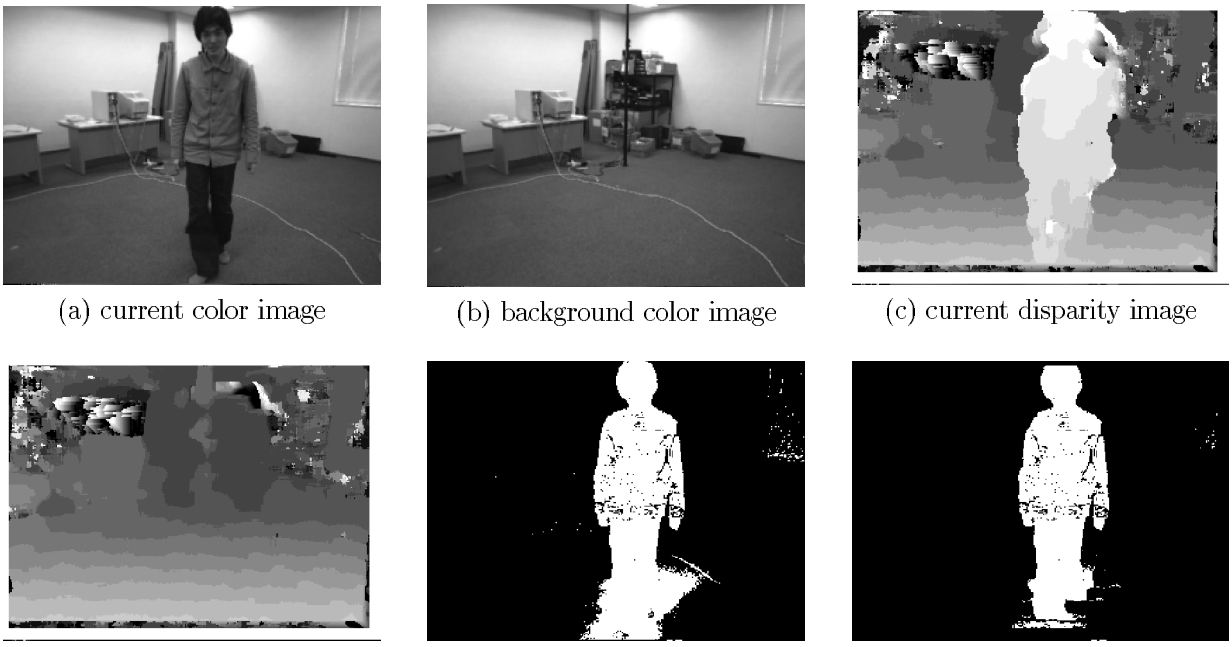
Before 3D shape reconstruction, we must generate silhouette images by background subtraction. Here, we describe the method of silhouette generation.

We use not only color information, but also disparity information taken by stereo method for robust silhouette generation. Only color information, light noise and shadows affects result images. Disparity images are strong for these effect, but we can get only prodigal silhouette. So we perform robust background subtraction by combining

these benefits. 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
      $\theta = \cos^{-1}\left(\frac{\mathbf{c_b}(c,y)\cdot\mathbf{c_c}(x,y)}{||\mathbf{c_b}(x,y)||||\mathbf{c_c}(x,y)||}\right)$
      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 a pixel condition of silhouette image, and th_D, th_U, th_L and th_A is disparity, upper(color), lower(color), angle threshold empirically defined. The result of silhouette generation is shown in Fig 11. Compare Fig 11(f) to Fig 11(e), shadow effect around the foot in Fig 11(e) is removed in Fig 11(f) because disparity image is not affected by the shadow.



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

(d) background disparity image (e) silhouette image without disparity image (f) silhouette image with disparity image

Figure 3. silhouette generation

4.2. Shape from Silhouette

With camera parameters and multiple view silhouette images, we perform shape from silhouette using octree data structure, to reconstruct object 3D shape. Perspective projection of an image generates a conic volume model(shown in Fig 4). Eventual 3D model is generated by combination of each conic 3D model. The equation of shape from silhouette is shown (2). And the image of shape from silhouette is shown Fig 5.

$$V_I = \bigcap_{i \in I} V_i \tag{2}$$

where, I is denoted as set of all silhouette images, and i is a image in set. And V_i is the object model generated from image i.

We don't use disparity images for reconstruction because they have only 8 level range data. Our volume of interest has 256^3 voxels.

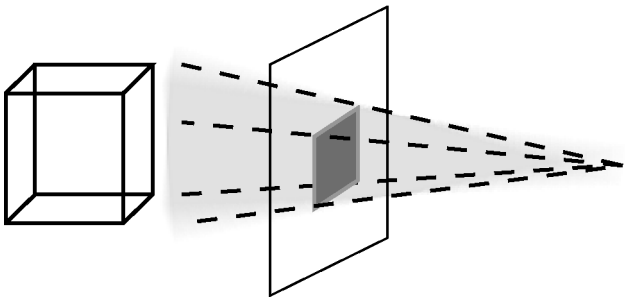


Figure 4. perspective projection

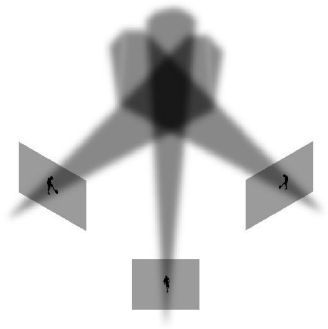


Figure 5. shape from silhouette

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 Fig 6, 7. 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 projecting them into all silhouette images, we perform the intersection of the projected cube region with silhouette region. Then, we describe the detail of the method.

At first, 8 vertices of "*universal cube*" are projected into all image planes, and search region is decided. Then, projected cube constitutes hexagon. But the main aim of our system is fast reconstruction, so we search the minimum rectangle including the hexagon. Next, we search pixels in the rectangle region, but don't search all pixels to shorten execute time. After decision of search rectangle, we perform intersection check. We show the intersection check in Fig 8.

If the region includes part of silhouette and background, the cube corresponding the region is temporarily determined as "*GRAY*" which means partly including the object. And if the all pixels of the search region are part of 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 while 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*". For this process, all temporary types are stored, and referred at child node to shorten execute time. For example, if the temporary parent node types are determined as "*WHITE*","*GRAY*","*GRAY*","*WHITE*", the final cube type is determined as "*GRAY*". In this case, the cube is

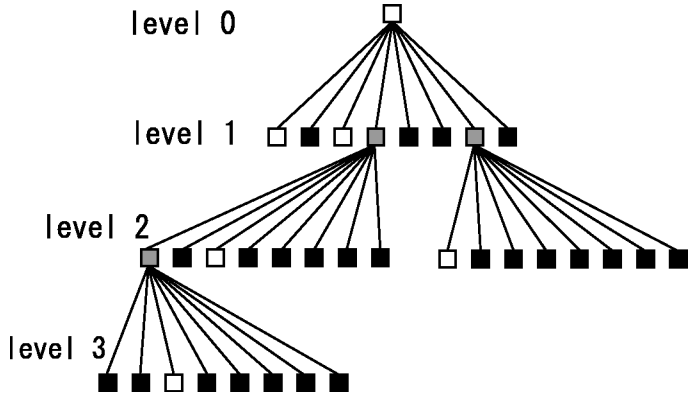


Figure 6. octree

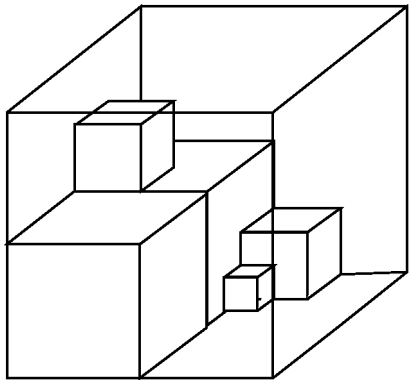
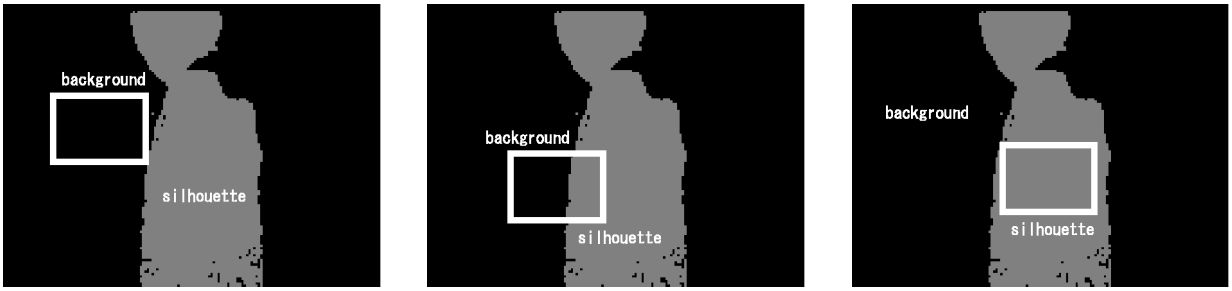


Figure 7. recursive cube subdivision

subdivided and have children, and the first and forth temporary nodes are certainly determined as "*WHITE*". 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 determined the type, we recursively iterate the same process. We show pseud code following.

```
for(i=0;i<cameraNumber;i++){
      refer the parent temporary cube type;
      if referred type is "GRAY" then
            project 8 vertices of the cube into image;
            decide minimum bounding rectangle including projected cube area;
            search the rectangle;
            decide the temporary cube type;
            if temporary cube type is "BLACK" then
                  store the temporary type as "BLACK";
                  break for loop;
            else if current cube type is "WHITE" or "GRAY" 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=0;j<8;j++)
            recursively iterate same process at each child node;
```



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

Figure 8. three types of intersection check. The white rectangle is search area.

4.4. Parallel operation

To reconstruct faster, we generate octree in parallel operation by the algorithm shown in Section 4.3. Generally, octree which has level h is generated in single process. In our method, we generate 8 octrees which level is $h-1$ in 8 parallel processes. These 8 processes correspond to subdivided 8 cubes. To separate processes in this way, 8 processes don't interfere each other. Run-time by this way corresponds to the longest time of 8 separated octree generation.

4.5. Internal voxels removal

After octree generation, we convert octree model to voxel model to display. Each octree node has 2^{3n} voxels. If we display these all 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.

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

At next step described in Section 4.6, we use only surface voxels.

4.6. Coloring of voxels

Generated 3D surface voxel model is not colored. In this step, voxel color is decided. First, we calculate the gravity g of the model as follows

$$\mathrm{g} = \frac{\sum_{\mathbf{v} \in SURFACE} \mathbf{v}}{n} \tag{3}$$

where, n is total number of surface voxels. The color of 3D surface model is decided depending on this g. The model is distributed into 4 parts which have a camera. A voxel in a part uses the allocated camera to color itself. A point which th voxel is projected into an image plane is calculated by Eqs. (1). The voxel color is decided this point color. Coloring of voxel can be found in Fig 9.

5. SYSTEM ARCHITECTURE

Fig 10 shows the architecture of the whole system. There are totally four cameras which are positioned spatially so that their field of view covered the volume of interest. The cameras are the Color Digiclops^{TM} (Point Grey Research, Inc), which have 3 cameras to perform stereo. The cameras are calibrated by the method described in Section 2. Each camera is connected to an individual PC. These cameras capture run-time images and generate disparity images, and perform silhouette generation. Although the camera generates disparity image, we do not use the disparity image for reconstructing the volume of the object, but only use it for generating the silhouette of the object. This is because that we intend to make the system run fast. If we apply a kind of volume merging process of the disparity images from the multiple viewpoint, then the calculation for the merging will make the system running speed slow. The camera images and silhouette images are sent to a host PC through the giga-bit ethernet. When reconstruction uses frame $n-1$, local PC captures frame n, and waits for the command from host PC to send the images. The host

computer is equipped with dual Pentium III 1GHz processors. Collecting all four color images and silhouette images, and perform 3D shape reconstruction, voxel conversion and internal voxels removal. The reconstructed 3D object is displayed using OpenGL.

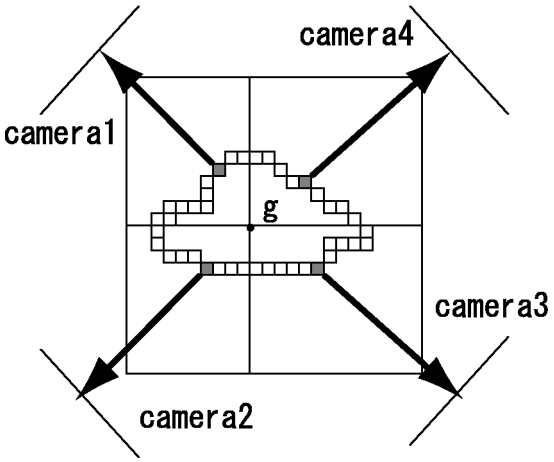


Figure 9. coloring of voxels seen from top view

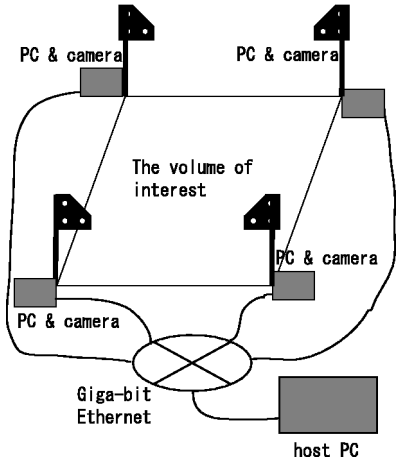


Figure 10. the system architecture

Table 1. The approximate calculation time for each step

Capture	50msec
Disparity image generation	100msec
Silhouette image generation	70msec
Image transfer	150msec
3D shape reconstruction with octree	150msec
Voxel conversion and internal voxel removal	20msec
Display	300 msec

6. RESULT

The system is used to reconstruct the object 3D shape with the following settings:

1. The volume of interest is a cube of size 2m x 2m x 2m.
2. The volume is divided into 256 x 256 x 256 voxels (size of a voxel is 7.8125mm).
3. Octree limited resolution is 8 level (size of a minimum cube equals a size of voxel).
4. The images are at a resolution of 320 x 240 pixels.
5. The color images are 24bit RGB color, and disparity images are 8bit Gray scale image.

Fig 11 shows example results obtained by the developed system. Fig 11(a)-(d) shows input images and Fig 11(e)-(h) shows the virtual view images that is generated from the reconstructed 3D shape. The table 1 shows the approximate calculation times for each step. The developed system also accepts previously captured image seqence as input. Images shown in Fig 12(a)-(b) are captured in a large hall[10] as shown in Fig. 13. Even from such images, the system can reconstruct the volume and render virtual view images as shown in Fig 12(c). It found some errors around the head in Fig 11(e)-(h) originated in some noises at silhouette generation. The silhouette images used for

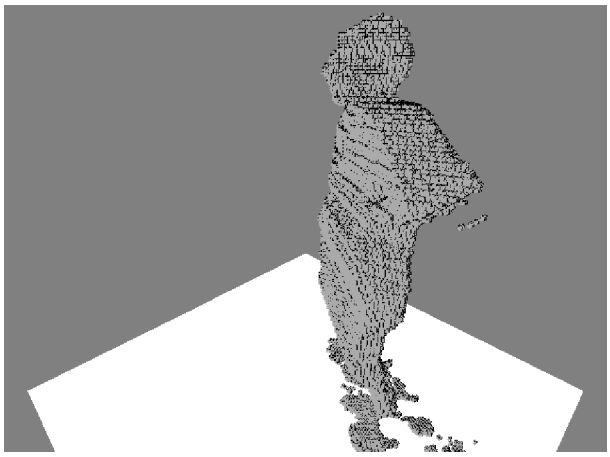
(a) input image1

(b) input image2

(c) input image3

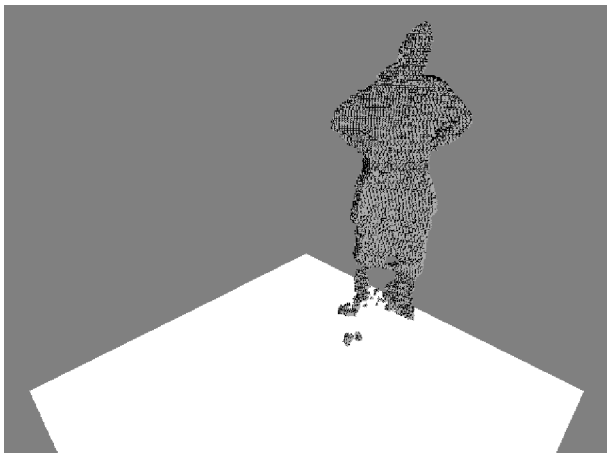
(d) input image4

(e) result image1



(f) result image2(no color)

(g) result image3



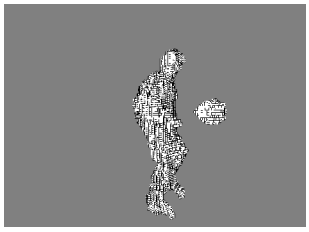
(h) result image4(no color)

Figure 11. input images and these results (on-line results)

(a) input image1

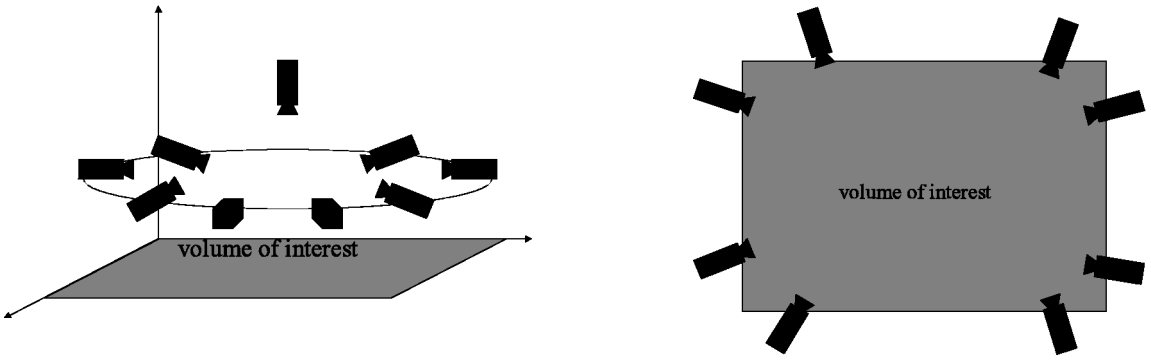
(b) input image2

(c) result image(color)



(d) result image(no color)

Figure 12. input images and these results (off-line results)



(a) position of cameras from side view (b) position of cameras from top view

Figure 13. camera position

this result have some holes around the head. A hole damages the model intensively. As the result of some holes in the silhouette images, the head of 3D model is shaved.

Some noises can be seen around the foot, too. Our method of silhouette generation which uses disparity images to narrow down the silhouette region sometimes cuts around the foot because the region of the foot and floor have same range level data in the disparity image. To avoid this problem, we refer the disparity value of the floor in advance and leave the region of floor.

The time of image transfer takes about 150 msec. At the current system, we send totally 32 x 320 x 240 x 4 bytes data per a capture. If we can merge silhouette images and color images, we can reduce time loss further.

Our system runs about 5 frames/sec without displaying the voxel surface model. However, if the total number of voxels is so large, the time of display takes long time because we render voxel model.

7. CONCLUSION AND FUTURE WORK

We proposed the fast 3D shape reconstructing system from multiple view images using Octree and silhouette, and show the result image of reconstructed 3D object shape. Our system consists of 4 cameras connected to individual PC and host PC to reconstruct. Each camera captures run-time image and generate silhouette image using disparity image. Host PC collects these silhouette images sent from each local PC and reconstruct 3D shape with octree. Reconstructed model is converted voxel surface model and displayed in arbitrary view. The bottle neck of our system is the time of displaying the model. Our system runs about 5 frames/sec without displaying the model. In present, total time from capture images to displaying the voxel surface model is over 500 msec if voxel data is so large. If we improve how to display the model, we render the object faster. It is useful for applications such as real time arbitrary view video system.

REFERENCES

1. A. Laurentini, "How many 2D silhouetts does it takes to reconstruct a 3D object?," *Computer Vision and Image Understanding* **67**, pp. 81–87, 1997.
2. H. Baker, "Three-dimensional modelling," *5th Int. Joint Conf. artif. Intell. '77* , pp. 649–655, 1977.
3. W. N. Martin and J. K. Aggarwal, "Volumetric descriptions of objects from multiple views," *IEEE Trans. Pattern Anal. Mach. Intell. PAMI-5* **2**, pp. 150–158, 1983.
4. M. Potmesil, "Generating octree models of 3D objects from their silhouettes in a sequence of images," *Computer Vision, Graphics, and Image Processing* **40**, pp. 1–29, 1987.
5. 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.
6. T. Wada, W. Xiaojun, S. Tokai, and T. Matsuyama, "Homography based parallel volume intersection: toward real-time volume reconstruction using active cameras," *Proceedings Fifth IEEE International Workshop on Computer Architectures for Machine Perception* , pp. 331–339, 2000.

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. lEEE Comput. Soc, Los Alamitos, CA, USA* **2**, pp. 714–729, 2000.
8. M. Okutomi and T. Kanade, "Multiple-Baseline Stereo," *lEEE Trans. Pattern Anal. Mach. Intell. PAMI-4* **15**, pp. 353–363, 1993.
9. R. Szeliski, "Rapid octree construction from image sequences," *CVGIP: Image Understanding* **58**, pp. 23–32, 1993.
10. S. Yaguchi and H. Saito, "Arbitrary view image generation from multiple silhouette images in projective grid space," *Proc. SPIE Videometrics and Optical Methods for 3D Shape Measurement* **4309**, pp. 294–304, 2001.