A 3D Modeling and Free-View Generation System Using Environmental Stereo Cameras

Hansung Kim,[1,2] Donghyun Kim,[1] Dongbo Min,[1] Kwanghoon Sohn[1]

[1] Digital Image Media Lab, Yonsei University, Seoul 120-749, Korea

[2] Centre for Vision, Speech and Signal Processing, University of Surrey, Surrey, UK

Received 1 February 2007; accepted 28 February 2008

ABSTRACT: We propose a 3D video system that uses environmental stereo cameras to display a target object from an arbitrary viewpoint. This system is composed of the following stages: image acquisition, foreground segmentation, depth field estimation, 3D modeling from depth and shape information, and arbitrary view rendering. To create 3D models from captured 2D image pairs, a real-time segmentation algorithm, a fast depth reconstruction algorithm, and a simple and efficient shape reconstruction method were developed. For viewpoint generation, the 3D surface model is rotated toward the desired place and orientation, and the texture data extracted from the original camera is projected onto this surface. Finally, a real-time system that demonstrates the use of the aforementioned algorithms was implemented. The generated 3D object can easily be manipulated, e.g., rotated or translated, to render images from different viewpoints. This provides stable scenes of a minimal area that made it possible to understand the target space, and also made it easier for viewers to understand in near real-time. © 2008 Wiley Periodicals, Inc. Int J Imaging Syst Technol, 17, 367–378, 2007; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/ima.20130

Key words: free-view generation; multiple video; 3D modeling; stereo analysis

I. INTRODUCTION

Two typical approaches can be taken when capturing visual information. As illustrated in Figure 1a, the first approach involves using a mobile camera manipulated by a cameraman (Ohta et al., 2002; Yamazoe et al., 2004). Although this approach allows the recording and transmitting of scenes with the least amount of data (i.e., a single video stream), it may lead to inconvenience for users, because the video data are captured from a subjective viewpoint. Viewers are able to watch only the provided scenes from passive positions. Another disadvantage of this approach is that the sway of the camera may confuse viewers.

The second approach is to use several cameras that are fixed in different locations in a given environment. Since these cameras provide objective and stable visual information from many viewpoints, it is easier for viewers to understand. However, an enormous amount of useless video must be captured to guarantee covering the entire area at all times. To determine the best position for observing an activity, it is necessary to switch between multiple videos. By increasing the number of cameras, this switching–monitoring operation may sometimes exceed the processing ability of the viewer.

To solve the aforementioned problems, we propose a 3D video system which can display a target object from an arbitrary viewpoint, as illustrated in Figure 1b. This system generates a 3D model of the space and target objects, and renders virtual views from any given viewpoint, according to where the viewer decides. The tasks required to fulfill this goal can be organized into a sequence of stages that process multiple images. Each of these stages presents us with its own problems. Specifically, some questions to be addressed in this article are as follows:

- Captured images are expressed in each sensor coordinate system so that data obtained from different viewpoints are expressed in different coordinate systems. How can we accurately express all the data in a single object-centered coordinate system?
- How can we reconstruct missing depth information from 2D images?
- Can we convert separate data obtained from each camera into a single surface description?
- How can we combine color information from the different viewpoints with surface geometry in order to render realistic images of the model from arbitrary viewpoints?

With recent progress in computer and video technologies, many computer vision-based 3D imaging systems have been developed. A shape-from-silhouette (SFS) technique is a very common way of converting silhouette contours into 3D objects (Matusik et al., 2002; Gross et al., 2003; Matsuyama et al., 2004; Yemeza and Schmitt, 2004). Silhouettes are readily and easily obtainable and

Correspondence to: Kwanghoon Sohn; e-mail: khsohn@yonsei.ac.kr

This research was supported by the MKE (Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the IITA (Institute of Information Technology Assessment) (IITA-2008-(C1090-0801-0011)), and was partially supported by the Ministry of Education, Science Technology (MEST), the Ministry of Knowledge Economy (MKE) and the Ministry of Labor (MOLAB) through the fostering project of the Lab of Excelency.

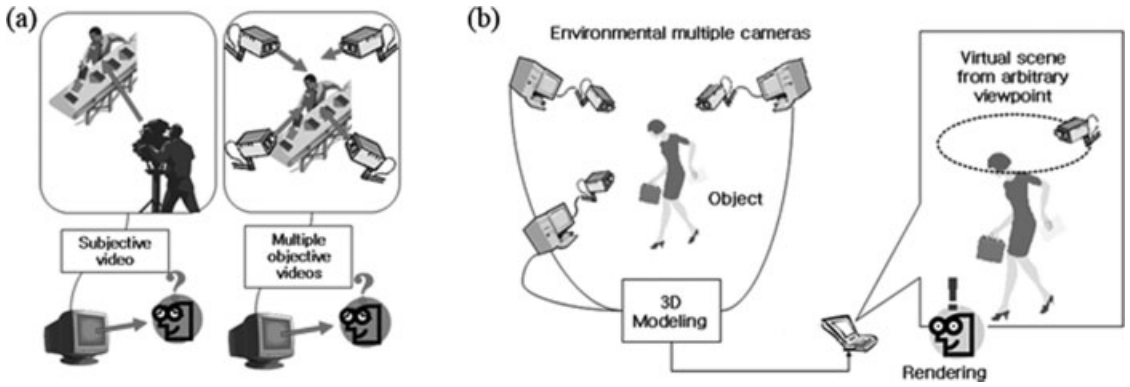

Figure 1. Capturing visual information: (a) Conventional approaches; (b) Proposed 3D video system.

the implementation of SFS methods is generally straightforward. The visual hulls constructed using SFS methods provide an upper bound on the shape of the object. This inherently conservative property is particularly useful in applications such as obstacle avoidance and visibility analysis. However, they have serious limitations when applied to concave surface regions or multiple, disjoint convex objects.

There has also been a lot of work on stereo vision for the recovery of dense scene structures from multiview images (Dhond and Aggarwal, 1989; Pulli, 1997; Esteban and Schmitt, 2002; Cheung et al., 2003). When monocular motion analysis and stereo vision are considered separately, each has its own inherent difficulties. Monocular motion analysis normally involves solving point correspondences, or nonlinear equations. Thus the computations are very sensitive to noise. Moreover, 3D motion interpretation is difficult because of structural ambiguities. On the other hand, stereo vision needs to solve the correspondence problem, i.e., matching features between given stereo image pairs. This problem, in general, is underdetermined. Other heuristics are also desirable. It is natural to consider integrating stereo and other properties in order to complement the performance of each.

In this article, we propose a new 3D modeling and free-view rendering system that uses both silhouette and disparity information to carve space with stereo cameras. The system aims to reconstruct the 3D structure of the target space from captured video streams by using fixed environmental stereo cameras. Scenes can then be generated by a virtual camera at an arbitrary position that employs a 3D video processing technique in real-time. It provides stable scenes of a minimal area in order to understand the target space. In turn, this makes it easier for viewers to understand the data.

In Section II, the proposed system, as well as a detailed description of the algorithms used in the system, is described. We propose a method of constructing the geometry of scenes and cameras. We also explain how to segment foreground regions in image sequences, how to solve correspondence problems and recover missing depth information in stereo image pairs, and how to reconstruct 3D models and generate dynamic scenes from arbitrary viewpoints. In Section III, we present and discuss the results obtained from the system. Section IV concludes by discussing some extensions of the algorithms and proposes possible directions for future work.

II. THE PROPOSED SYSTEM

The system comprises two subsystems: object segmentation and disparity estimation in the capturing PCs, and 3D modeling and rendering in the 3D modeling servers. This is shown in Figure 2.

When target objects are captured by cameras, each capturing PC segments the objects and estimates the disparity fields, then it transmits the segmented masks, disparity fields, and color textures of the objects to a 3D modeling server via User Datagram Protocol. The modeling server then generates 3D models of each object from the gathered masks and disparity fields, and tracks each object in a sequence of 3D model frames. Finally, the server generates a video at the designated point of view with the 3D model and texture information.

The following subsections describe in detail the algorithms used by the system.

A. Camera Calibration. Camera calibration refers to determining the values of the extrinsic and intrinsic parameters of the camera. The key idea behind calibration is to write projection equations linking the known coordinates of a set of 3D points and their projections, and determine the camera parameters. To find the coordinates of some 3D points, camera calibration methods rely on one or more images of calibration patterns, that is, a 3D object of known geometry, possibly located in a known position and generating image features which can be located accurately.

Generally, there are two approaches to camera calibration with known geometry. The first method directly recovers the intrinsic and extrinsic parameters, and the second method [introduced by Trucco and Verri (1998)] estimates the projection matrix first, without solving explicitly for the various parameters, which are then computed as closed-form functions of the entries of the projection matrix. The proposed system uses the second method because it is simpler than the first method, and a projection matrix is used

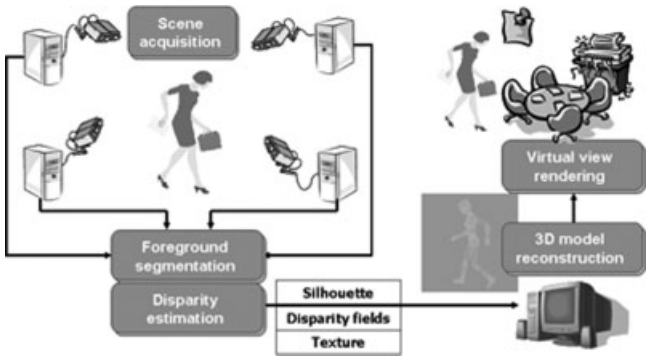

Figure 2. Overall configuration of the proposed system.

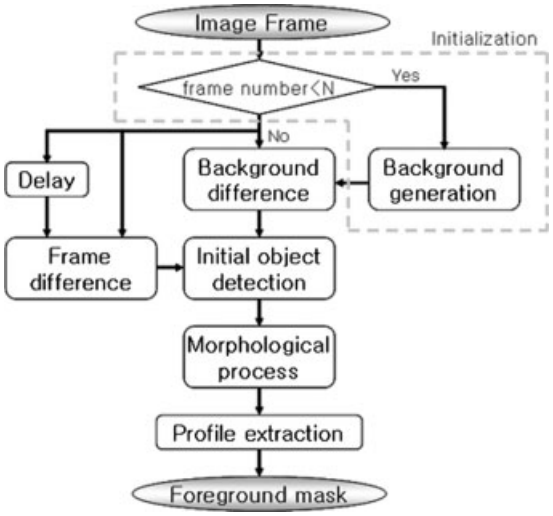

Figure 3. Block diagram of the proposed segmentation algorithm.

directly to reconstruct the 3D model. The projection matrix of each camera is extracted by using the calibration patterns and the least square techniques, and then the following camera parameters are calculated from the projection matrices.

Projection matrix:

$$P = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix}$$

Extrinsic parameters:

$$3 \times 3 \text{ rotation matrix } R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

$$\text{3D translation vector } t = [\, t_x \quad t_y \quad t_z \,]^{\mathrm{T}}$$

Intrinsic parameters:

Lengths of effective pixel size units s_x and s_y. Image center coordinates c_x and c_y. Focal length f.

B. Foreground Segmentation. Real-time foreground segmentation is one of the most important components of the proposed system, since segmentation performance decides the quality of the final 3D model. We segment foreground regions using background subtraction and interframe difference.

Figure 3 shows the process of the proposed foreground segmentation algorithm. At first, the background masks $I_{\min}(x,y)$ and $I_{\max}(x,y)$ are modeled with the minimum and maximum intensities of the first N frames, respectively, because the background information is very sensitive to noise and change of illumination. Then, the frame difference mask $I_{\mathrm{fd}}(x,y)$ is calculated by the difference between two consecutive frames. In the third step, an initial foreground mask is constructed from the frame difference and the background difference masks by the OR process, that is, if a pixel of the current frame satisfy one of the conditions in Eq. (1), it is determined to belong to an initial foreground region. $\mathrm{Th_{tol}}$ and $\mathrm{Th_{fd}}$ mean threshold values for the background and frame difference regions, respectively.

$$\begin{aligned} I_{\mathrm{cur}}(x,y) &< I_{\min}(x,y) - \mathrm{Th_{tol}} \\ I_{\mathrm{cur}}(x,y) &> I_{\max}(x,y) + \mathrm{Th_{tol}} \\ I_{\mathrm{fd}}(x,y) &> \mathrm{Th_{fd}} \end{aligned} \tag{1}$$

However, because of camera noise and irregular object motion, there are also noise regions in the initial mask. A conventional way of eliminating noise regions is using a morphological operation to filter out small regions. Therefore, we refine the initial mask by a closing process and eliminate small regions with a region-growing technique.

Finally, to smooth the boundaries of the foreground and to eliminate holes inside the regions, we propose a profile extraction technique which is improved from the single human profile extraction algorithm proposed by Kumar et al. (2000). A weighted one pixel thick drape is moved from one side to the opposite side. The adjacent pixels of the drape are connected by an elastic spring that covers the object but does not infiltrate into gaps whose widths are smaller than a threshold M. This process is performed from all quarters and the region wrapped by four drapes denotes the final foreground region. Figure 4 shows the profile extraction process as applied to an initial object.

C. Disparity Estimation. The most important problem in realizing 3D imaging systems is reconstructing the 3D coordinates of a captured scene. Thus far, many active and passive methods have been proposed to recover depth information from real scenes. Active techniques utilize ultrasonic or lasers to illuminate the work space, so that they yield fast and accurate depth information (Feiner et al., 1993; Iddan and Yahav, 1994). However, there are limitations to these techniques with respect to measurement range and hardware cost. Conversely, passive techniques based on computer vision are less sensitive to environmental conditions and typically require a simpler and less expensive setup for range sensing. Those approaches are capable of estimating depth information from acquired images and camera parameters (Kanade et al., 1996; Fehn et al., 2002).

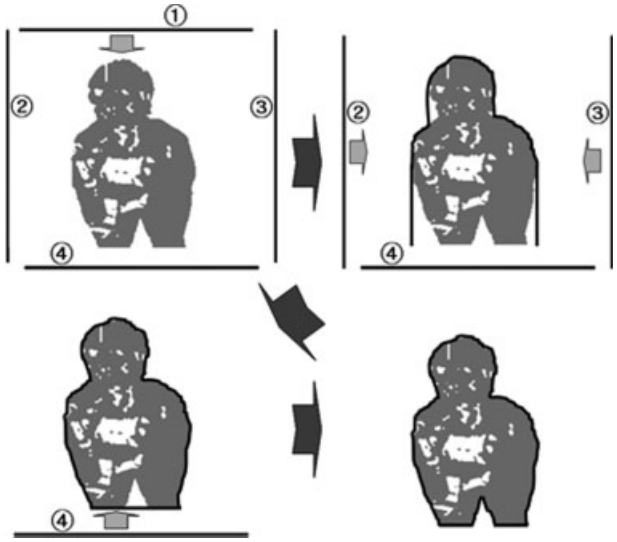

Figure 4. Profile extraction.

A problem in depth estimation with passive techniques is to find the corresponding pairs (I_1 and I_2) of a single world point w in two separate image views. If we assume that the cameras are identical and the coordinate systems of both cameras are aligned in parallel, the determination of the disparity from I_1 to I_2 becomes finding a function $d(x,y)$:

$$I_2(x, y) = I_1(x + d(x, y), y). \tag{2}$$

Once the disparity of the corresponding pair is extracted, we can get the distance to the point as follows, where f is the focal length of the camera and B is the baseline distance.

$$Z = \frac{fB}{d} \tag{3}$$

For these and other reasons, considerable effort has been expended on the disparity estimation problem since the 1970s. Recently, Scharstein and Szeliski (2002) discussed the taxonomy of existing stereo algorithms, and Brown et al. (2003) reviewed advances in correspondence methods, methods for occlusion, and real-time implementation. However, most of these methods have serious limitations when they are applied to common applications since many kinds of 3D imaging systems require real-time calculation of disparity fields for dynamic scenes. Most real-time systems have been implemented by special purpose hardware, such as Digital Signal Processors or Field Programmable Gate Arrays (Fehn et al., 2002; Darabiha et al., 2003). With increasing clock speeds and development of Graphics Processing Unit (GPU), real-time stereo processing has recently been realized on common desktop computers (Mühlmann et al., 2002; Yang and Pollefeys, 2005; Mairal et al., 2006). However, these computers generally show poor quality in wide-ranging applications since they use simple area-based algorithms such as Sum of Absolute Difference.

We previously proposed a two-stage algorithm to find smooth and precise disparity vector fields in a stereo image pair (Kim et al., 2004). The algorithm consisted of dense disparity estimation and edge-preserving regularization. This resulted in a clean disparity map with good discontinuity localization, but the computational cost proved to be so high that it is impractical in real-time. Therefore, this algorithm was improved into a fast disparity estimation algorithm using the results of segmentation (as discussed in the previous section). This new algorithm assumes that a stereo camera set does not move, and there is no moving object for a few seconds in an initialization step for generating background information. Accurate and detailed disparity information for the background region is estimated in advance, and then only the disparities of the moving foreground regions are calculated and merged into the background disparity fields. Figure 5 shows a block diagram of the fast disparity estimation algorithm.

C.1. Background Disparity Estimation. Dense disparity fields of the background regions are initially estimated in a hierarchical way. The first step in hierarchical estimation is a $B \times B$ block-based initial disparity estimation. In the second step, dense disparity vectors for each pixel are estimated based on the initial block vectors. To cover all the probable disparity candidates, nine initial vectors (one from the current block and eight from neighboring blocks) are tested within a small search range α from the vector. To improve computational efficiency in disparity estimation, we use a region-dividing technique (Kim et al., 2004). This technique performs point matching in the order of the possibility of correct matching and divides the region into subregions at the true matching point. After the region split into two subregions during the matching process, the search ranges of the points in each subregion are restricted to the corresponding subregion.

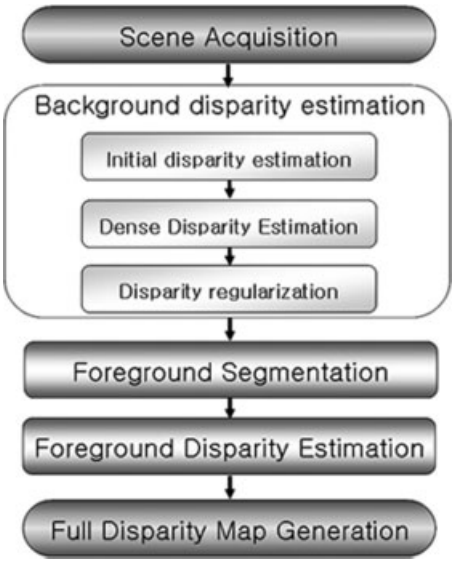

Figure 5. Block diagram of the fast disparity estimation algorithm.

The disparity vectors estimated by the aforedescribed method provide generally reliable information. However, the spatial correlation of the estimated vector fields is not considered. The disparity fields of the background are estimated only once at an initialization step, so we refine the fields in the continuous domain by regularization in order to provide more detailed and reliable background disparity fields. The energy functional consists of a fidelity term and a smoothing term:

$$E(d) = \int_\Omega (I_l(x, y) - I_r(x + d(x, y), y))^2 \, dx \, dy + \lambda \int_\Omega \psi(\nabla d(x, y), \nabla I_l(x, y)) \, dx \, dy, \tag{4}$$

where Ω is an image plane, λ a weighting factor of the smoothing term, and $\Psi(\bigtriangledown d, \bigtriangledown I_l)$ a potential function whose gradient is given by

$$\nabla(\psi(\nabla d, \nabla I_l)) = \frac{1}{(1 + \nabla I_l^2)^2} \nabla d. \tag{5}$$

The minimization problem is addressed by solving the associated Euler–Lagrange equation, and the following corresponding asymptotic state of the parabolic system.

$$\frac{\partial d}{\partial t} = \lambda \, \mathrm{div}\left(\frac{1}{(1+\nabla I_l^2)^2} \nabla d(x, y)\right) + (I_l(x, y) - I_r(x + d, y)) \frac{\partial I_r(x + d, y)}{\partial x} \tag{6}$$

This partial differential equation (PDE) corresponds to the nonlinear diffusion equation with an additional reaction term. $1/(1 + \bigtriangledown I^2)^2$ is a diffusivity function which plays the role of a discontinuity marker. Therefore, the diffusion process leads to a disparity vector map with smooth continuous surfaces and preserves its discontinuities at the object boundaries.

To solve Eq. (6), we discretize the parabolic system by using the finite differences, and find the regularized disparity field in a recursive manner by updating the field using Eq. (7).

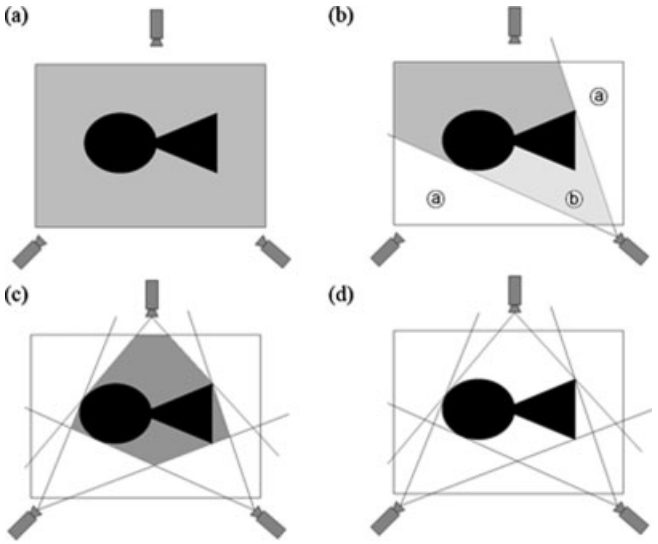

Figure 6. Shape from silhouette and disparity: (a) Whole volume containing objects; (b) Carving by the first camera; (c) Result by SFS; (d) Result by the proposed technique.

$$\frac{d^{k+1}(x,y)-d^{k}(x,y)}{\tau}=\lambda\left\{\begin{array}{l}\frac{\partial}{\partial x}\left[g\left[\left|\frac{\partial I_{\mathrm{l}}(x,y)}{\partial x}\right|^{2}\right]\times\frac{\partial d^{k}(x,y)}{\partial x}\right]\\+\frac{\partial}{\partial y}\left[g\left[\left|\frac{\partial I_{\mathrm{l}}(x,y)}{\partial y}\right|^{2}\right]\times\frac{\partial d^{k}(x,y)}{\partial y}\right]\end{array}\right\}$$
$$+\left(I_{\mathrm{l}}(x,y)-I_{\mathrm{r}}(x+d^{k}(x,y),y)\right)\times\frac{\partial I_{\mathrm{r}}(x+d^{k}(x,y),y)}{\partial x}$$
$$+\left(d^{k}(x,y)-d^{k+1}(x,y)\right)\times\left(\frac{\partial I_{\mathrm{r}}(x+d^{k}(x,y),y)}{\partial x}\right)^{2}\qquad(7)$$

C.2. Foreground Disparity Estimation. The most important requirement of foreground disparity estimation is processing speed. This is because the fields must be updated in every frame in a real-time system. Therefore, we use a simplified algorithm for the foreground regions with additional constraints. Hierarchical disparity estimation in the previous section is applied to the blocks which include the foreground regions. Initial search ranges are also restricted by the neighbor background disparities since foreground objects always exist in front of the background region. Equation (8) shows the search range decision where $\mathrm{SR_{Max}}$ and $\mathrm{SR_{Min}}$ refer to the maximum and minimum search ranges, respectively, and d_{ln} and d_{rn} are the left and right neighboring background disparities of the foreground region on the same scanline.

For L → R disparity

$$\mathrm{SR_{Max}}=\mathrm{Min}(\mathrm{d_{ln,}}\,d_{\mathrm{rn}}).\qquad(8)$$

For R → L disparity

$$\mathrm{SR_{Min}}=\mathrm{Max}(\mathrm{d_{ln,}}d_{\mathrm{rn}}).$$

As a result, the search ranges are restricted by three factors: background disparities, the region-dividing technique, and hierarchical estimation. Thus, the processing time of foreground estimation is greatly reduced.

In background disparity estimation, wrong disparities around the boundary regions are corrected by energy-based regularization. However, in general, the regularization process requires such high computational complexity that it cannot be applied to foreground estimation. Moreover, segmentation errors may cause errors around the borders of the foreground in the estimation. Therefore, we check the reliability of the disparity for the pixels in the boundary blocks which included the boundary between the background and foreground regions. The final disparities of the pixels in the boundary blocks are determined by the following conditions, where d_{fore} is an estimated disparity and d_{back} is a disparity of the background region at the same position.

If $(|I_{\mathrm{r}}(x,y)-I_{\mathrm{l}}(x+d_{\mathrm{fore}}(x,y),y)|\,|I_{\mathrm{r}}(x,y)-I_{\mathrm{l}}\ (x+d_{\mathrm{back}}(x,y),y)|)$

$$d_{\mathrm{final}}(x,y)=d_{\mathrm{fore}}(x,y)\qquad(9)$$

else

$$d_{\mathrm{final}}(x,y)=d_{\mathrm{back}}(x,y)$$

D. 3D Modeling. The transmitted data from the capturing PCs are used to reconstruct a 3D model of the objects. The estimated disparity fields are converted into depth information with the camera parameters, and a 3D model of the object is then reconstructed from the silhouette and disparity fields. Figure 6 shows a simple concept of the shape from silhouette and disparity technique on 2D plane. There is a working volume in Figure 6a and the object which we are trying to reconstruct is only known to be somewhere inside of it. Figure 6b shows the carving result using the data from the first camera. If we use only silhouette information, region ⓐ is carved but region ⓑ still remains. By using depth information we can carve the region ⓒ. Figures 6c and 6d show the carving result by conventional SFS technique and the proposed technique, respectively. As it is shown, the result by SFS is much coarser than that by the proposed algorithm.

Based on Pulli's method (1997), we propose a simple volume carving algorithm using silhouette and disparity with the camera parameters at the same time, as shown in Figure 7. All voxels $M(X, Y, X)$ in the 3D spaces are projected onto multiple images I_n using the following equation, where P_n is the projection matrix of camera C_n.

$$[u,v,s]^{\mathrm{T}}=P_n[X,Y,Z,1]^{\mathrm{T}}\qquad(10)$$

If the projected points of M are included in the foreground region of the image, then we check if the voxel places behind the range of depth calculated from the disparity. d_{voxel}, the distance from the

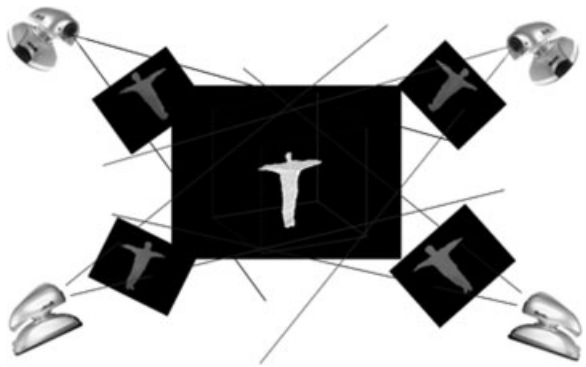

Figure 7. 3D modeling with a shape from silhouette and disparity.

camera to point M, and d_{surf}, the distance from the camera to the surface, are calculated by Eqs. (11) and (12). f is the focal length of the camera extracted as a intrinsic camera parameter, B is the baseline distance between the lenses of the camera, $d(u,v)$ is the disparity of the projected point $I(u,v)$, and t_x, t_y, and t_z are translation of the camera in extrinsic camera parameters.

$$d_{\text{voxel}} = \sqrt{(X_M - t_x)^2 + (Y_M - t_y)^2 + (Z_M - t_z)^2} \qquad (11)$$

$$d_{\text{surf}} = -\frac{fB}{d(u,v)} \qquad (12)$$

If all projected points of M are included in the foreground region and behind the distance to the surface of multiple images, we select the point as the inside voxel of the object. The pseudo-code for the function is given as

Modeling funtion()

for (all points in the modelling space)

count = 0

for($t = 0$; $t <$ number of cameras ; $t++$)

calculate$[\text{u, v, s}]^{\text{T}} = P[X, Y, Z, 1]^{\text{T}}$

$u = u/s$

$v = v/s$

if($m(u,v)$ is included in the shape)

calculate d_{voxel} and d_{surf}

if($d_{\text{voxel}} \geq d_{\text{surf}}$)

count++

if(count = number of cameras)

$M(X,Y,Z)$ is inside the model

else

$M(X,Y,Z)$ is outside the model

Testing all the points in a 3D model is, however, a very time-consuming process and results in heavy data. Therefore, we use an octree data structure for modeling. For each voxel of a given level, 27 points (i.e., each corner and the centers of the edges, faces, and a cube) are tested. If all checking points are either included in or excluded from an object, the voxel is assigned as a full or empty voxel, respectively. Otherwise, the voxel is split into eight subvoxels and is tested again at the next refinement level. Figure 8 shows the structure of the octree. This structure dramatically reduces the modeling speed and the amount of data.

E. Virtual View Rendering. In the rendering stage, a virtual view of the objects is synthesized from the 3D model and texture information. The proposed system employs the projective texture mapping method (Everitt, 2001). This mapping method projects the texture image onto the 3D objects like a slide projector, as shown in Figure 9. Consequently, the resolution of the texture image is retained during the texture mapping process, regardless of the resolution and the shape of the mapped 3D model. Moreover, this method can be implemented as an OpenGL functional library. This makes it possible to take advantage of a high-speed graphic accelerator. By merging the working space (background), which was modeled in advance, a complete 3D model of the working space and object is reconstructed.

Finally, the scenes at viewpoints requested by users are generated. This system provides the following modes of controlling the viewpoint of a virtual camera.

Tracking mode. In this mode, the virtual camera observes the object from a position above and behind the target object. The direction of movement is estimated by tracking a global path of objects from the movements in the previous consecutive frames.

Orbiting mode. We can make the virtual camera go around the object like a satellite when the target object stops in one position. Thus, the orbiting mode makes it possible to observe blind (self-occluded) spots.

User-control mode. We can set and move the position of the virtual camera manually so that we can observe the target object from any point of view.

Stereo mode. Two frames are consecutively rendered at viewpoints which are 6.5 cm away (in the horizontal direction) from each other. This produces scenes much like those captured by a virtual stereo camera. The rendered scenes are shown with a sense of reality on a stereo display.

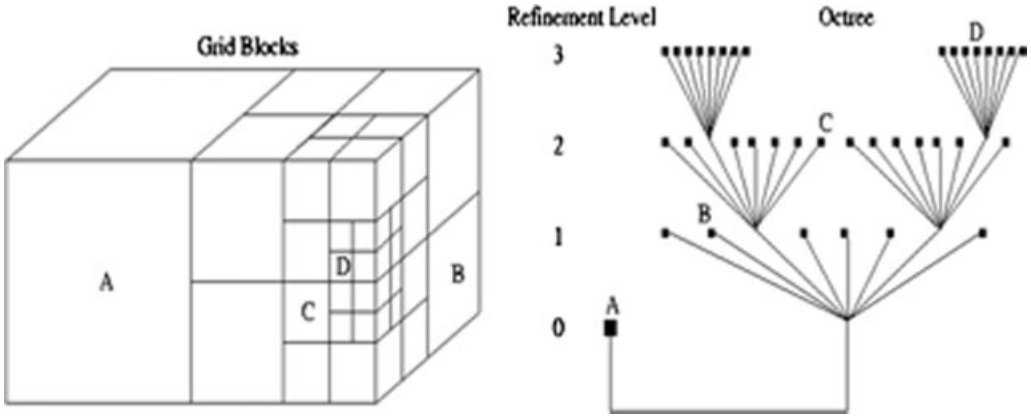

Figure 8. Octree structure.

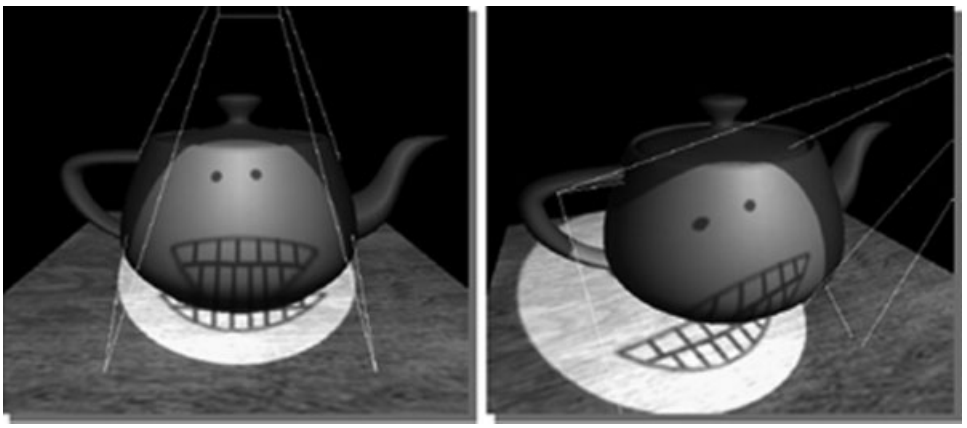

Figure 9. Projective texture mapping (Everitt, 2001).

III. EXPERIMENTAL RESULTS

Figure 10 shows the layout of the experimental studio. We implemented a distributed system using five PCs and four stereo cameras (two Digiclops® and two Bumblebee® cameras; http://www.ptgrey.com/). The systems were realized with commercially available hardware. Four Pentium IV 3.0-GHz PCs were used to capture the video streams, segment objects, and estimate the disparity fields linked with each camera. The modeling PC contained a Pentium IV CPU and a Quadro FX1300 graphic accelerator.

A. Camera Calibration. To extract the camera parameters of the four stereo cameras, we implemented the 2D pattern-based camera calibration technique based on the projection matrix as discussed in Section II.A. A chessboard with 40 corners derived from 0.12 m × 0.12 m was used for calibration. We took two shots at different board positions where the world coordinates were known. We then calculated the projection matrix by solving the homogeneous linear system with the Single Value Decomposition (Trucco and Verri, 1998). Corners were detected by the functions *cvFindChessBoardCornerGuesses( )* and *cvFindCornerSubPix( )* provided in the OpenCV open library (http://www.opencv.com/).

We tested the accuracy of the matrices by projecting other world points to the image plane by computing the pixel coordinates (u, v) of the points using the estimated projection matrices. Then, we computed the error mean between the computed (u, v) by the projection matrices and detected (u, v) by corner detection. We tested 40 points within 1.5 m from the center of the world coordinates for each matrix. Table I shows the average errors in pixels for each camera and matrix. The average errors were less than 1 pixel in the horizontal (u) direction and 0.5 pixel in the vertical (v) direction.

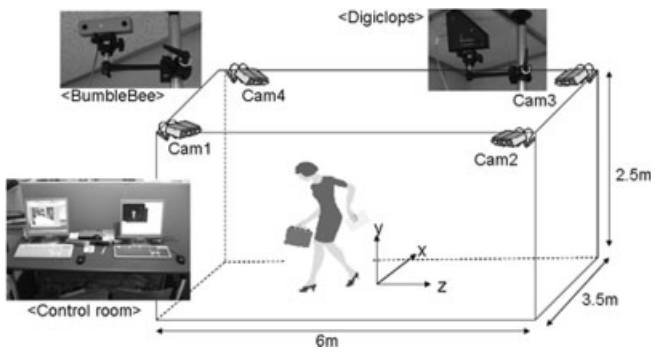

Figure 10. Experimental room.

From the projection matrices, we calculated all the camera parameters including the intrinsic and extrinsic parameters. The projection matrices and parameters were used in depth reconstruction, 3D modeling, and the rendering process.

B. Depth Estimation. To verify the accuracy of the reconstructed depth information from the estimated disparity fields, we tested the algorithm for the scene with clearly known depth information in a virtual environment. Table II shows the parameters used in the simulation, and Figure 11 shows the setup of the objects, the stereo images captured by the virtual cameras, and an estimated disparity map of the left image using the background estimation algorithm.

From the estimated disparity fields, we reconstructed the depth information for each point using Eq. (3) since an ideal parallel stereo camera was used. The evaluation of accuracy for the particular points in Figure 11a is shown in Table III, and Table IV shows the root-mean-square-error (RMSE) of the estimated depth of each object. In Table IV, the RMSE is calculated for the inside pixels of the objects, except at the border of two pixels. When the boundary pixels were included, the RMSE increased considerably because of errors around the object boundaries. More research concerning boundary detection and accurate disparity estimation at the regions is clearly needed. The RMSE was stepped up as the object receded in the distance. As the distance between the cameras and the object increased, depth errors, caused by disparity errors, were increasingly amplified since depth are inversely proportional to disparity, as shown in Eq. (3). For example, a 0.1 pixel disparity error at 3 m leads to a depth error of 0.017 m, but the same disparity error at 10 m leads to a 0.195 m depth error. As a result, the accuracy of depth reconstruction depends on the distance to the object as well as the

Table I. Average errors of the estimated projection matrices

	Average Projection Errors (pixels)	
Camera	Horizontal	Vertical
Cam1	0.98445	0.55464
Cam2	0.91357	0.44827
Cam3	0.57490	0.46410
Cam4	0.77927	0.45470

Table II. Parameters used in simulation

Stage	Parameter	Values
Virtual camera	Focal length	35 mm
	Baseline distance	176 mm
	Pixel size	0.1165 mm
Foreground segmentation	Background generation	$N = 50$
	Background difference	$\mathrm{Th}_{\mathrm{tol}} = 10$
	Frame difference	$\mathrm{Th}_{\mathrm{fd}} = 5$
Disparity estimation	Block size	$B = 8$
	Dense disparity range	$\alpha = 2$
Disparity regularization	Lagrange multiplier	$\lambda = 2000$
	Time step size	$\tau = 0.0001$
	Number of iterations	$T = 150$

accuracy of the disparity fields. However, this can be overcome by adjusting the camera parameters.

The proposed algorithm was then applied to a set of stereoscopic sequences captured by the Digiclops camera, which provides a rectified stereo sequence at a speed of 30 frames/s.

Figure 12 shows several frames of the resulting sequences; the left column shows the left images, the middle column shows the segmented foregrounds, and the right column shows the final disparity fields. The image sequences were captured in a typical office environment without any special lighting equipment or any arrangement of objects.

In the segmentation results, we can see that most moving objects are segmented without holes. However, some parts of the background are included in the foreground when the object moves fast or when two objects overlap in a scene because of frame differences and profile extraction, respectively. Once in a while, infiltration of the background into an object is also observed. In the results of the final disparity fields, a 3D structure of the scene can be imagined.

C. 3D Modeling. To verify the proposed shape from silhouette and disparity algorithm, models from the same data set with the SFS algorithm and the proposed algorithm were compared. The segmentation and disparity information from each camera were captured at a resolution of 640 × 480 pixels and the 3D space was modeled at a resolution of 256 × 256 × 256 on a 1 cm × 1 cm × 1 cm voxel grid.

Figure 13 shows the captured images and Figure 14 shows the segmentation masks with the disparity fields. Segmentation in this test was performed in a semimanual way to confirm the contribution of the disparity fields to modeling. Figures 15a and 15c show models generated by the SFS method. We can see that the final models generated by the SFS method are very coarse and bulky, especially in the Z-direction. This is because the camera distance between Cam1 and Cam2 was larger than that between Cam2 and Cam3, as shown in the experimental setup depicted in Figure 10. On the other hand, Figures 15b and 15d show models generated by the proposed method. The models look more natural because the redundancies in the models produced by the SFS method are removed by the depth information. However, the wrong disparity fields do not take any effect in refining the models. They sometimes go even as far as damaging the models. We can see that the right arm of the model is thinner than the left arm in Figure 15b, and the left leg appears unnatural in Figure 15d.

D. Real-Time System and Arbitrary View Rendering. A real-time 3D modeling system was realized using the proposed techniques. In this system, the resolution of each camera was reduced to 320 × 240 pixels since the computational cost dramatically increases as the resolution of the images increases. Table V shows a run-time analysis with the proposed system. The times listed are the average processing times when one man moves in the experimental space.

The modeling and rendering process ideally takes about 16 frames/s. However, in practical systems when working with camera clients, the process also needs idle time in order to get the full information that is necessary to make 3D models from all clients at a speed of 7–8 frames/s. The modeling speed theoretically cannot exceed the speed of the camera client. The frame rate of the whole system also depends on the complexity of the objects.

It is difficult to compare the computational efficiency with other methods precisely because they are executed under different

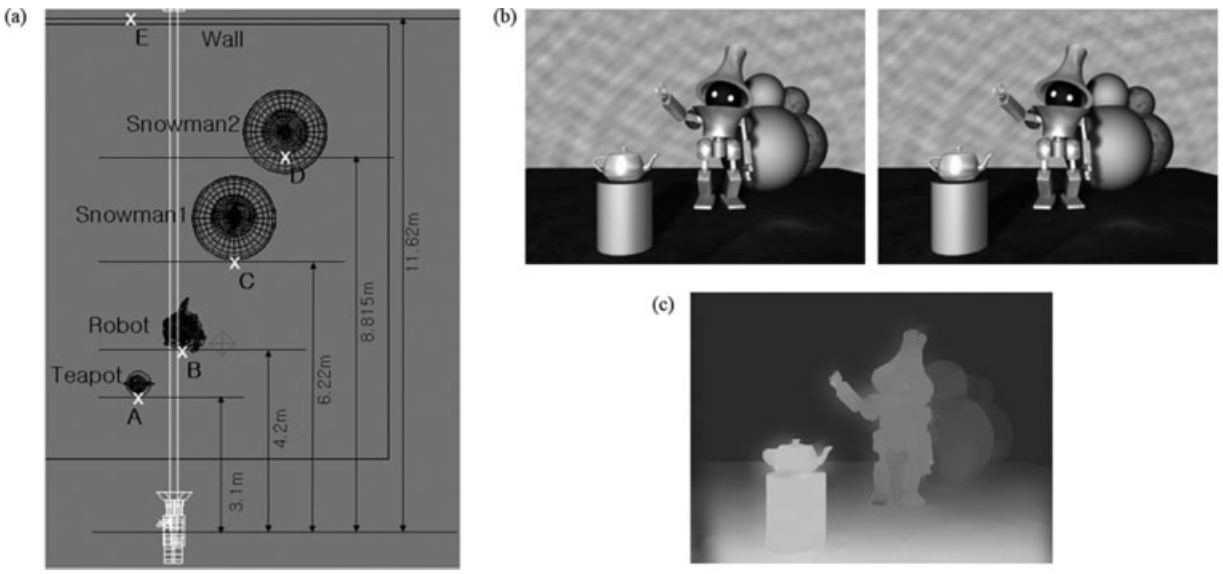

Figure 11. System setup and simulation results. (a) Setup for simulation; (b) Stereo pair; (c) Estimated disparity field.

Table III. Depth of the particular points shown in Figure 11a

Point	Estimated Disparity (Pixel)	Estimated Depth D_E (m)	True Depth D_T (m)	Error $D_T - D_E$ (m)
A	17	3.1453	3.11	−0.0353
B	12.6	4.2316	4.22	−0.0116
C	8.4	6.3297	6.22	−0.1097
D	6.2	8.5633	8.815	−0.2517
E	4.7	11.2851	11.78	0.4949

conditions. According to referenced papers of SFS methods, the system by Matsuyama et al. (2004) shows a speed of 6 frames/s with 24 PCs by distributed processing, and the system by Cheung et al. (2003) works in 5 frames/s. Comparing with these systems, the running time of the proposed system shows good performance in processing speed.

Figure 16 shows snapshots of a rendered 3D model in various rendering modes. We selected the nearest camera from a given viewpoint and projected the texture from it to the model using projective texture mapping. Therefore, in the rear part of the model as seen from the selected camera, the textures appear distorted and cause unnaturalness. However, the system generally renders proper scenes.

IV. CONCLUSION AND FUTURE WORKS

We have presented a complete 3D imaging system using multiple stereo cameras. The system begins with estimating the geometry of real objects and finally displays realistic images of those objects from arbitrary viewpoints in real-time.

Table IV. RMSE of depth for each object in Figure 11

Object	RMSE (m)
Teapot	0.0367
Robot	0.0856
Snowman 1	0.1268
Snowman 2	0.1857

Stages of this system include image acquisition, foreground segmentation, depth field estimation, 3D modeling from depth and shape information, and arbitrary view rendering. The generated 3D object can easily be manipulated, e.g., rotated or translated, to render images from different viewpoints. The system provides stable scenes that enable viewers to understand the activities of users in near real-time. The system has the following advantages:

1. *Speed*. The set of algorithms produce one of the fastest running times of all current algorithms, and the overall speed of the full system works at a speed of 8 frames/s. The speed adapts both to the size of the images and the number of input cameras.
2. *Robustness*. The system works without problems for the majority of different scenes since it relies mainly on the quality of both foreground segmentation and the disparity estimation algorithms.
3. *Low cost*. No active devices (for example, 3D range scanners) need to be used; only a few stereo digital video cameras. Moreover, the systems are realized with common PCs used in everyday life. The cost of the stereo cameras is still high, but these can also be substituted by two normal cameras with external rectification functions.

Figure 12. Test sequence and simulation results.

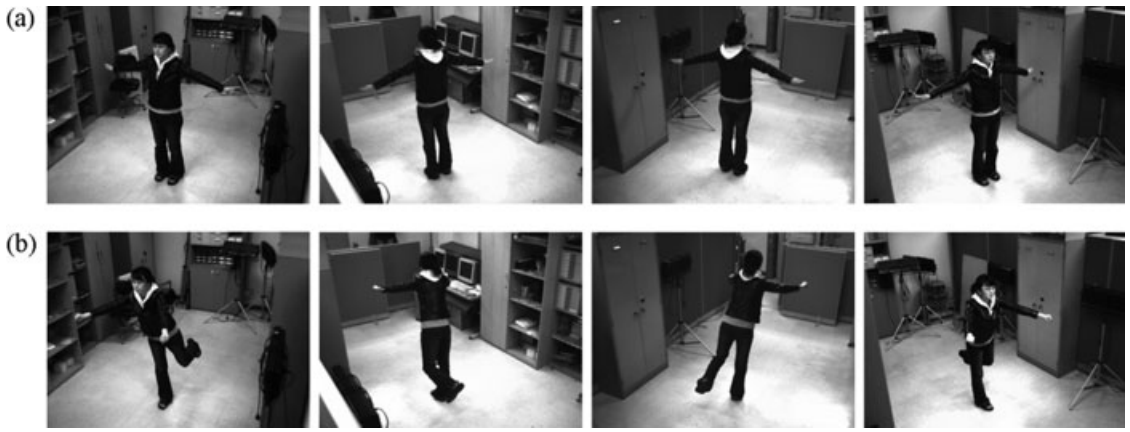

Figure 13. Captured images: (a) Set 1; (b) Set 2.

4. *Quality*. Although the proposed system gives only an approximate voxel-based geometry, the quality of the reconstructed models is apparently better than the quality achieved when using previous approaches.

Although significant advances have been made in this article, there are still many areas that need to be explored in future research.

In terms of camera calibration, it is necessary to consider the intercamera calibration between each camera in order to increase efficiency and the accuracy of the calibration. This will result in more accurate depth reconstruction and 3D modeling. Several intrinsic parameters such as radial distortion and skew parameters were not considered in this article. Several previous approaches have already proposed methods of estimating these parameters (Tsai, 1987; Zhang, 1999). Further consideration of the parameters would help improve the overall accuracy of the system.

We have to develop a more powerful segmentation algorithm robust to variations in lighting conditions and shadows. The performance of the segmentation decides the efficiency and quality of the final disparity fields as well as the quality of the model reconstructed by the silhouette information. Especially, when foreground regions are classified into background regions because of wrong segmentation, this causes serious errors in the final disparity fields since the fields are not updated.

Future work on disparity estimation will follow two main directions. Since most parameters are determined experimentally, parameter optimizing techniques are needed to render the system stable and efficient. In particular, time step size τ and the number of iterations T must be carefully decided. A large τ value can speed up the algorithm, but may lead to a divergence in the PDE solution. On the other hand, a small τ and a large T will slow the computational time for the algorithm without improving the quality of the disparity fields. A statistical analysis of input images as a preprocessing step may help in finding the optimal parameters. Another perspective of future work will be to improve the accuracy of the disparity fields at the object boundary regions. The proposed regularization technique still blurs the disparity fields around the object boundaries though it uses an edge-preserving regularization technique. The ambiguous fields around the regions can damage the final model in 3D reconstruction.

We developed a simple volume carving algorithm for 3D modeling. However, the reconstructed 3D model is still rough when rendering natural virtual scenes. Fundamentally, a voxel-based structure is not a very good approximation for the object surface since the mesh obtained from the voxel is too dense, and the orientations of the mesh faces suffer from severe quantization as they are strictly aligned with the coordinate axes. In future work, we will examine a mesh optimization process. A marching cube algorithm and an energy optimization method may reveal possible solutions

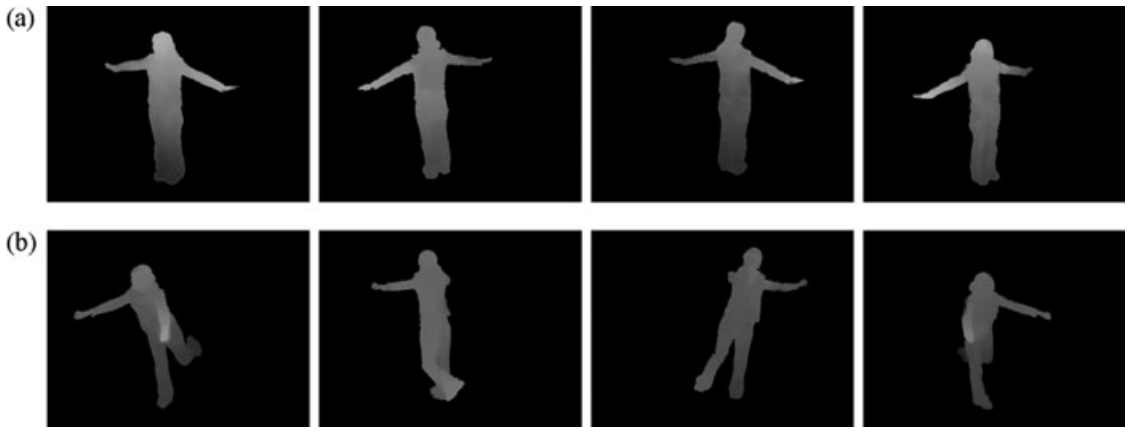

Figure 14. Extracted shape and disparity: (a) Set 1; (b) Set 2.

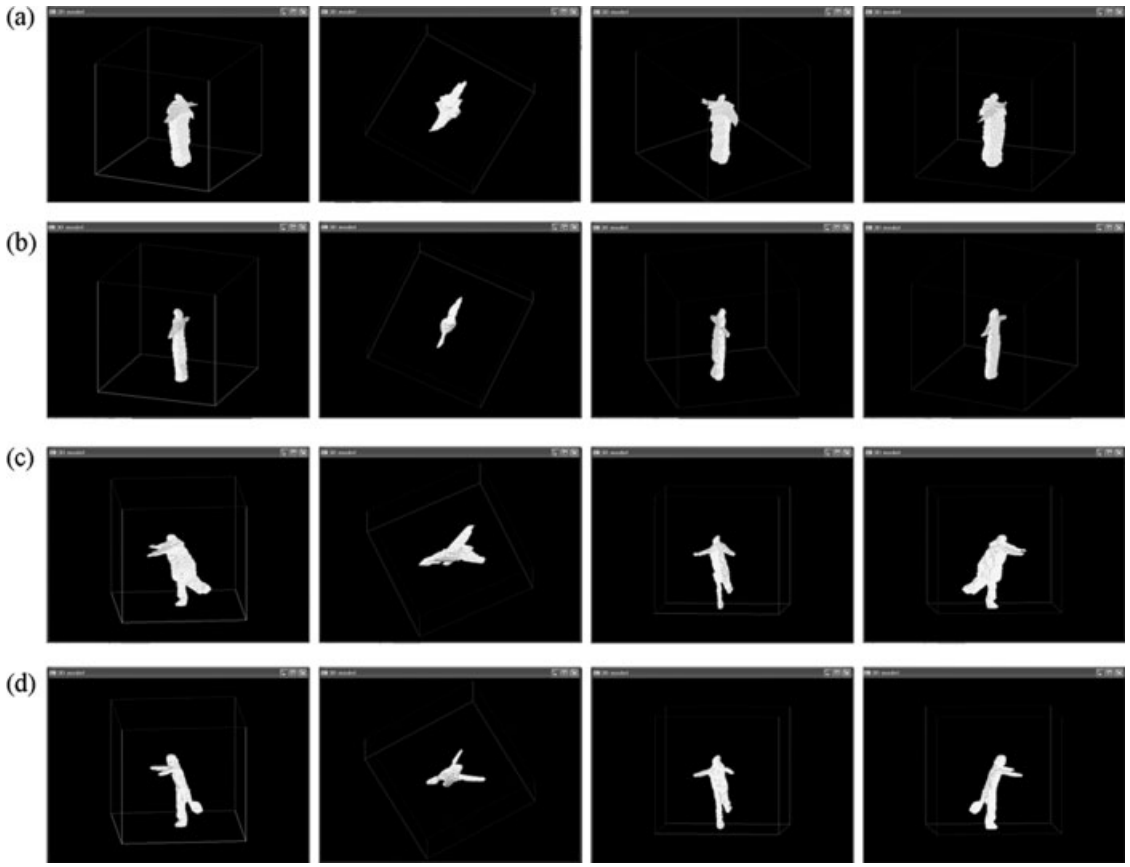

Figure 15. Results of 3D modeling: (a) Set 1 by conventional SFS algorithm; (b) Set 2 by the proposed algorithm; (c) Set 2 by conventional SFS algorithm; (d) Set 2 by the proposed algorithm.

Table V. Processing speed analysis of the system

Camera Clients		3D Modeling and Rendering	
Function	Time (ms)	Function	Time (ms)
Capturing	28.26	Initialization	12.00
Segmentation	19.53	3D Modeling	23.59
Disparity estimation	17.81	Tracking	0.24
Transmission	1.25	Rendering	26.30
Total time	*66.85*	*Total time*	*62.13*
Frames/second	*14.96 f/s*	*Frames/second*	*16.10 f/s*

(Lorensen and Cline, 1987; Pulli, 1997). Secondly, the overall quality of the output video stream should be improved. The resolution of the final rendered scene can be improved if we can use the 640 $\times$ 480 video streams at 30 frames/s. Moreover, the number of video streams can be increased in order to improve the quality. All of these goals can be achieved by using hardware accelerators such as GPU (Akbarzadeh et al., 2006; Mairal et al., 2006) and by employing further optimizations to the implementation.

Virtual view rendering was not a central part of this research, since we only used a simple projective texture mapping method.

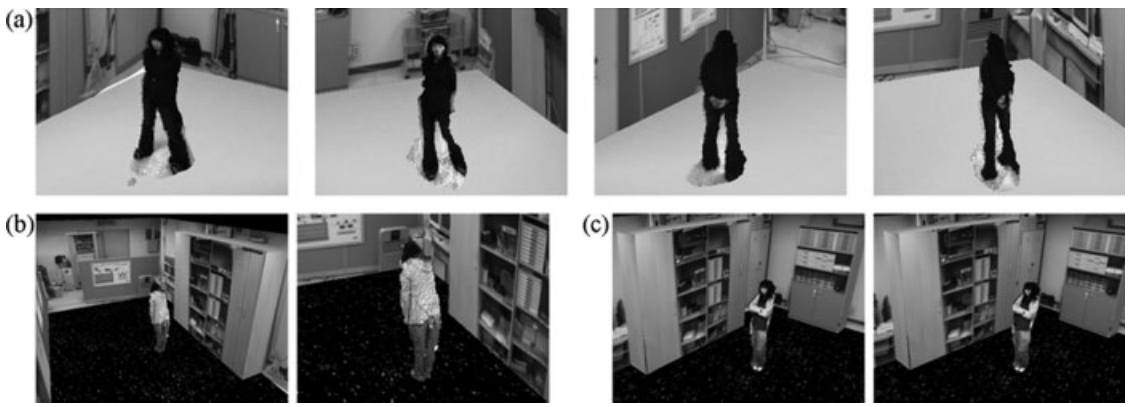

Figure 16. Rendered scenes: (a) Orbiting mode; (b) Zooming mode; (c) Stereo mode.

We projected a texture of the nearest camera from the viewpoint, so it produced coarse rendered scenes especially in the rear region from the selected camera. To improve the quality of rendering, a view-dependent texturing technique should be developed.

Although the system still has problems to be overcome, we believe that the proposed algorithms and the system set a framework for many future algorithms and applications.

REFERENCES

A. Akbarzadeh, J.M. Frahm, P. Mordohai, B. Clipp, C. Engels, D. Gallup, P. Merrell, M. Phelps, S. Sinha, B. Talton, L. Wang, Q. Yang, H. Stewenius, R. Yang, G. Welch, H. Towles, D. Nister, and M. Pollefeys, Towards Urban 3D Reconstruction From Video, Proc 3DPVT, 2006, pp. 1–8.

M. Brown, D. Burschka, and G.D. Hager, Advances in computational stereo, IEEE Trans PAMI 25 (2003), 993–1008.

G.K.M. Cheung, S. Baker, and T. Kanade, Visual hull alignment and refinement across time: A 3D reconstruction algorithm combining shape-from-silhouette with stereo, Proc CVPR, 2003, pp. 375–382.

A. Darabiha, J. Rose, and W.J. MacLean, Video-rate stereo depth measurement on programmable hardware, Proc CVPR, 2003, pp. 203–210.

U. Dhond and J. Aggarwal, Structure from stereo: A review, IEEE Trans Syst Man Cybern 19 (1989), 1489–1510.

C.H. Esteban and F. Schmitt, Multi-stereo 3D object reconstruction, Proc 3DPVT, 2002, pp. 159–167.

C. Everitt, Projective texture mapping, NVIDIA SDK White Paper, 2001.

C. Fehn, E. Cooke, O. Schreer, and P. Kauff, 3D analysis and image-based rendering for immersive TV applications, Signal Process Image Commun 17 (2002), 705–715.

S. Feiner, B. Maclntyre, and D. Seligmann, Knowledge-based augmented reality, Commun ACM 36 (1993), 53–62.

M. Gross, S. Würmlin, M. Naef, E. Lamboray, C. Spagno, A. Kunz, E. Koller-Meier, T. Svoboda, L. Van Gool, S. Lang, K. Strehlke, A. Vande Moere, and O. Staadt, Blue-c: A spatially immersive display and 3D video portal for telepresence, Proc ACM SIGGRAPH, 2003, pp. 819–827.

G.J. Iddan and G . Yahav, 3D imaging in the studio (and elsewhere. . .), Proc SPIE 4298 (1994), 48–55.

T. Kanade, A. Yoshida, K. Oda, H. Kano, and M. Tanaka, A stereo machine for video-rate dense depth mapping and its new applications, Proc CVPR, 1996, pp. 196–202.

H. Kim, Y. Choe, and K. Sohn, Disparity estimation using region-dividing technique with energy-based regularization, Opt Eng 43 (2004), 1882–1890.

P. Kumar, K. Sengupta, and S. Ranganath, Real time detection and recognition of human profiles using inexpensive desktop cameras, Proc ICPR, 2000, pp. 1096–1099.

W.E. Lorensen and H.E. Cline, Marching cubes: A high resolution 3D surface construction algorithm, Proc ACM SIGGRAPH, 1987, pp. 163–169.

J. Mairal, R. Keriven, and A. Chariot, Fast and efficient dense variational stereo on GPU, Proc 3DPVT, 2006, pp. 97–104.

T. Matsuyama, X. Wu, T. Takai, and T. Wada, Real-time dynamic 3-D object shape reconstruction and high-fidelity texture mapping for 3-D video, IEEE Trans CSVT 14 (2004), 357–369.

W. Matusik, H. Pfister, A. Ngan, P. Beardsley, R. Ziegler, and L. McMillan, Image-based 3D photography using opacity hulls, Proc ACM SIGGRAPH, 2002, pp. 427–437.

K. Mühlmann, D. Maier, J. Hesser, and R. Männer, Calculating dense disparity maps from color stereo images, an efficient implementation, Int J Comput Vis 47 (2002), 79–88.

Y. Ohta, Y. Sugaya, H. Igarashi, T. Ohtsuki, and K. Taguchi, Share-Z: Client/server depth sensing for see-through head-mounted displays, PRESENCE 11 (2002), 176–188.

K. Pulli, Surface reconstruction and display from range and color data, Ph.D. Thesis, University of Washington, Seattle, WA, 1997.

D. Scharstein and R. Szeliski, A taxonomy and evaluation of dense two-frame stereo correspondence algorithms, Int J Comput Vis 47 (2002), 7–42.

E. Trucco and A. Verri, Introductory techniques for 3-D computer vision, Prentice Hall, New Jersey, 1998.

R.Y. Tsai, A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, IEEE J Robot Automat RA-3 (1987), 323–344.

H. Yamazoe, A. Utsumi, N. Tetsutani, and M. Yachida, Vision-based human motion tracking using head-mounted cameras and fixed cameras for interaction analysis, Proc ACCV, 2004, pp. 682–687.

R. Yang and M. Pollefeys, A versatile stereo implementation on commodity graphics hardware, Real-Time Imaging 11 (2005), 7–18.

Y. Yemeza and F. Schmitt, 3D reconstruction of real objects with high resolution shape and texture, Image Vis Comput 22 (2004), 1137–1153.

Z. Zhang, Flexible camera calibration by viewing a plane from unknown orientations, Proc ICCV, 1999, pp. 666–673.