REAL-TIME GENERATION AND HIGH FIDELITY VISUALIZATION OF 3D VIDEO

T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara

{tm,wxj,takesi-t,nob}@vision.kuee.kyoto-u.ac.jp
Graduate School of Informatics, Kyoto University
Sakyo, Kyoto, 606-8501, Japan

ABSTRACT

3D video is the ultimate image media recording dynamic visual events in the real world as is. Recorded object actions can be observed from any viewpoint; 3D video records the object's full 3D shape, motion, and precise surface properties (i.e. color and texture). In this paper, we present research attainments so far obtained: 1. a PC cluster system for real-time reconstruction of dynamic 3D object action from multi-viewpoint video images, 2. a deformable 3D mesh model for reconstructing the accurate 3D object shape, and 3. an algorithm of rendering natural-looking texture on the 3D object surface from the multi-viewpoint video images. Experimental results demonstrate the effectiveness of these methods in generating high fidelity object images from arbitrary viewpoints.

1. INTRODUCTION

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

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

To cultivate the 3D video world and make it usable in everyday life, we have to solve the following technical problems:

- *Computation Speed:* We have to develop both faster machines and algorithms, because near frame-rate 3D shape reconstruction has been attained only in coarse resolution and moreover texture mapping onto the reconstructed 3D shape is still done off-line.
- *High Fidelity:* To obtain high fidelity 3D video in the same quality as ordinary video images, we have to develop high fidelity texture mapping methods as well as increase the resolution.
- *Wide Area Observation:* 3D areas observable by the systems developed so far are confined to about $2m \times 2m \times 2m$, which should be extended considerably to capture human actions like sports playing.
- *Data Compression:* Since naive representation of 3D video results in huge data, effective compression methods are required to store and transmit 3D video data[6].
- *Editing and Visualization:* Since editing and visualization of 3D video are conducted in the 4D space (3D geometric + 1D temporal), we have to develop human-friendly 3D video editors and visualizers that help a user to understand dynamic events in the 4D space[7].

This paper first describes our second generation PC cluster system for reconstructing dynamic 3D object action from multi-viewpoint video images, by which a temporal series of 3D voxel representations of the object action can be obtained in real-time. The improvements from the first system[3] rest in the introduction of 1. IEEE 1394 digital cameras to increase the video quality, 2. 25 active cameras to realize wide-area active observation, 3. 30 PCs to facilitate real-time processing. Several results of quantitative performance evaluation are given to demonstrate its effectiveness. Then, we present a deformable 3D mesh model for reconstructing the accurate 3D object shape and an algorithm of rendering video texture on the reconstructed dynamic 3D object surface from the multi-viewpoint video images. Experimental results demonstrate the effectiveness of these methods in generating high fidelity object images from arbitrary viewpoints.

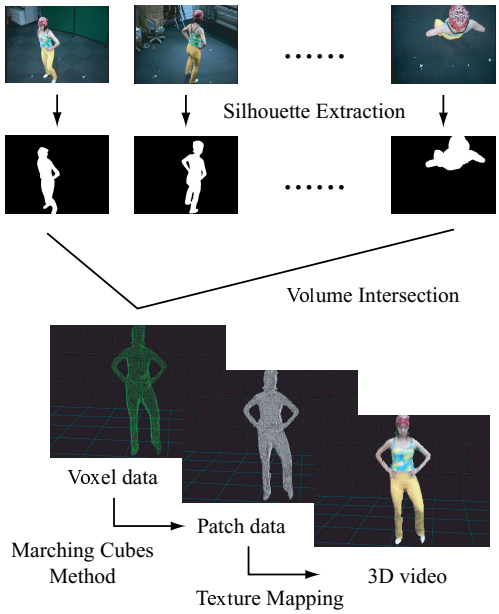


Figure 1: 3D video generation process

2. BASIC SCHEME OF 3D VIDEO GENERATION

Figure 1 illustrates the basic process of generating a 3D video frame in our system:

1. **Synchronized Multi-Viewpoint Image Acquisition**: A set of multi-viewpoint object images are taken simultaneously by a group of distributed video cameras (top row in Figure 1).

2. **Silhouette Extraction**: Background subtraction is applied to each captured image to generate a set of multi-viewpoint object silhouettes (second top row in Figure 1).

3. **Silhouette Volume Intersection**: Each silhouette is back-projected into the common 3D space to generate a visual cone encasing the 3D object. Then, such 3D cones are intersected with each other to generate the voxel representation of the object shape (third bottom in Figure 1).

4. **Surface Shape Computation**: The discrete marching cubes method[8] is applied to convert the voxel representation to the surface patch representation. Then the generated 3D mesh is deformed to obtain accurate 3D object shape(second bottom in Figure 1).

5. **Texture Mapping**: Color and texture on each patch are computed from the observed multi-viewpoint images (bottom in Figure 1).

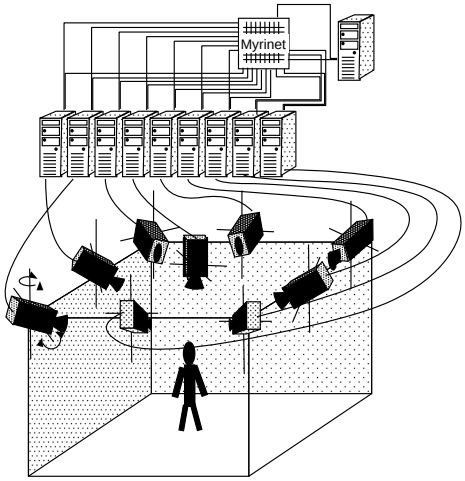


Figure 2: PC cluster for real-time active 3D object shape reconstruction system.

By repeating the above process for each video frame, we have a live 3D motion picture.

In the following sections, we describe our real-time 3D shape reconstruction system, deformable mesh model, and high fidelity video texture mapping algorithm.

3. REAL-TIME DYNAMIC 3D OBJECT SHAPE RECONSTRUCTION SYSTEM

3.1. System Organization

Figure 2 illustrates the hardware organization of our real-time *active* 3D object shape reconstruction system. It consists of

- PC cluster: 30 node PCs (dual Pentium III 1GHz) are connected through Myrinet, an ultra high speed network (full duplex 1.28Gbps). PM library for Myrinet PC clusters[9] allows very low latency and high speed data transfer, based on which we can implement efficient parallel processing on the PC cluster.

- Distributed active video cameras: Among 30, 25 PCs have Fixed-Viewpoint Pan-Tilt (FV-PT) cameras[10], respectively, for active object tracking and image capturing. In the FV-PT camera, the projection center stays fixed irrespectively of any camera rotations, which greatly facilitates real-time active object tracking and 3D shape reconstruction. Moreover, digital IEEE 1394 video cameras are employed to enhance video quality (image size: 640×480).

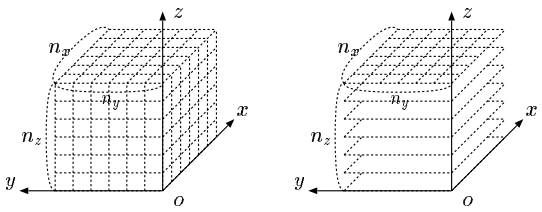


Figure 3: 3D shape representations: (a) 3D voxel space, (b) parallel plane space.

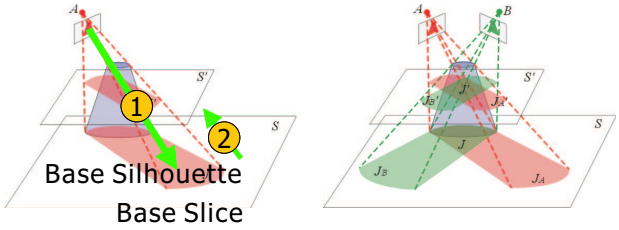


Figure 4: Plane-based volume intersection method

3.2. Real-time Plane-Based Parallel Pipeline Volume Intersection Method

In the naive volume intersection method, the 3D space is partitioned into small voxels (Figure 3(a)), and each voxel is mapped onto each image plane to examine whether or not it is included in the object silhouette.
To realize real-time 3D volume intersection, we proposed in [3]

1. the plane-based volume intersection algorithm to improve the computational efficiency,
2. a parallel processing method using the PC cluster, and
3. the introduction of pipeline processing to further increase the processing speed.

The plane-based volume intersection algorithm consists of the following processes:

1. Partition the 3D space into a set of equally spaced parallel planes (Figure 3(b)).
2. Project the object silhouette observed by each camera onto the base plane (Figure 4 left, ①).
3. Project each base silhouette onto the other planes (Figure 4 left, ②).
4. Compute 2D intersection of all silhouettes projected on each plane (Figure 4 right).

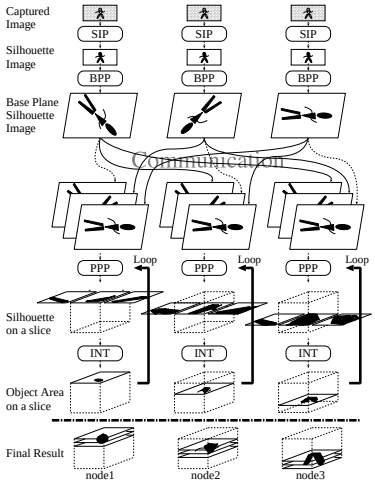


Figure 5: Processing flow of the parallel pipeline 3D shape reconstruction.

5. Stack up the intersected silhouettes to get 3D object shape.

From a computational point of view, this algorithm has the following advantages [3] :

- The plane-to-plane homographic projection is more computationally efficient than the 3D perspective projection (Figure 4 left, ①). Moreover, we developed a very efficient computational method to implement the plane-to-plane projection, which greatly accelerates the computation speed.
- The projection between parallel planes can be realized by simple 2D affine transformation (Figure 4 left, ②), which also accelerates the computation.
- The projection from an image plane onto the base plane and the silhouette intersection on a plane can be done independently of the others, which enables us to introduce efficient data-level parallel processing.

Figure 5 illustrates the processing flow of the parallel pipeline 3D shape reconstruction. It consists of the following five stages:

1. *Image Capture :* Triggered by a capturing signal, each PC with a camera captures a video frame (top row in Figure 5).
2. *Silhouette Extraction :* Each PC with a camera extracts an object silhouette from the video frame (second top row in Figure 5).
3. *Projection to the Base-Plane :* Each PC with a camera projects the silhouette onto the common base-plane in the 3D space (third top row in Figure 5).

4. *Base-Plane Silhouette Duplication :* All base-plane silhouettes are duplicated across all PCs over the network so that each PC has the full set of all base-plane silhouettes (forth row in Figure 5). Note that the data are distributed over all PCs (i.e. with and without cameras) in the system.

5. *Silhouette Intersection :* Each PC computes 2D silhouette intersection on its specified parallel planes respectively(three bottom rows in Figure 5).

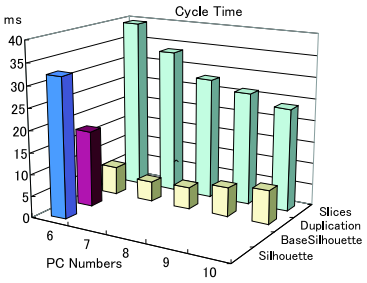
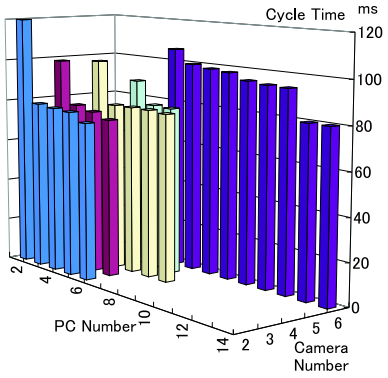
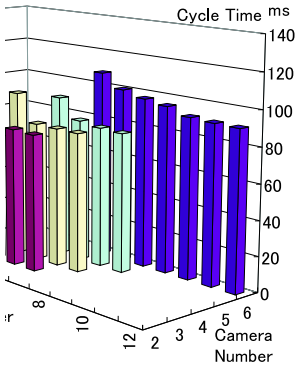


Figure 6: Average computation time for each pipeline stage measured by changing the number of PCs.



(a) Soft Trigger



(b) Hard Trigger

Figure 7: Overall computation time for reconstructing one 3D shape measured by changing the numbers of cameras and PCs.

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

3.3. Performance Evaluation

In the experiments of the real-time 3D volume reconstruction, we used 6 digital IEEE1394 cameras placed at the ceiling (like in Figure 2) for capturing multi-viewpoint video data of a dancing human. We will discuss their synchronization method later. The size of input image is 640×480 pixels. We measured the time taken to reconstruct one 3D shape in the voxel size of 2cm$\times$ 2cm$\times$ 2cm contained in a space of 2m $\times$ 2m $\times$ 2m.
In the first experiment, we analyzed processing time spent at each pipeline stage by changing the number of PCs from 6 to 10. Figure 6 shows the average computation time [1] spent at four pipeline stages: “Silhouette Extraction”, “Projection to the Base-Plane”, “Base-Plane Silhouette Duplication” and “Silhouette Intersection”. Note that the image capturing stage is not taken into account in this experiment, which will be discussed later.
From this figure, we can observe the followings:

- The computation time for the *Projection to the Base-Plane* stage is about 18ms, which proves the plane-to-plane projection algorithm proposed in [3] is very efficient.

- With 6 PCs (i.e. with no PCs without cameras), the bottleneck for real-time 3D shape reconstruction rests at the *Silhouette Intersection* stage, since this stage consumes the longest computation time (i.e. about 40ms).

- By increasing the number of PCs, the time taken at that most expensive stage decreases considerably (i.e. well below 30ms). This proves the proposed parallelization method is effective. Note that while the time spent at the Base-Plane Silhouette Duplication stage increases as the number of PCs is increased, it stays well below 30ms.

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

- With more than 8 PCs, we can realize real-time (video-rate) 3D shape reconstruction.

In the second experiment, we measured the total throughput of the system including the image capturing process by changing the numbers of cameras and PCs. Figure 7 shows the throughput[2] to reconstruct one 3D shape.
In our PC cluster system, we developed two methods for synchronizing multi-viewpoint video capturing: use an external trigger generator (**Hard Trigger**) and control the cameras through network-communication (**Soft Trigger**). The performance was evaluated for the two methods in Figure 7 (a) and (b), respectively.
From Figure 7 we can get the following observations:

- In both synchronization methods, while the throughput is improved by increasing PCs, it saturates at a constant value in all cases: for Soft Trigger about $75 \sim 80$ ms and for Hard Trigger about $80 \sim 90$ms.

- Comparing the hard and soft triggers, the latter shows a slightly better performance.

Since as was proved in the first experiment, the throughput of the pipelined computation itself is about 30ms, the elongated overall throughput is due to the speed of *Image Capture* stage. That is, although a camera itself can capture images at a rate of 30 fps individually, the synchronization reduces its frame rate down into half. This is partly because the external trigger for synchronization is not synchronized with the internal hardware cycle of the camera and partly because it takes some time to transfer image data to PC memory.
In summary, the experiments proved the effectiveness of the proposed real-time 3D shape reconstruction system: the plane-based volume intersection method, its acceleration algorithm, and the parallel pipeline implementation. Moreover, the proposed parallel processing method is enough flexible to scale up the system by increasing numbers of cameras and PCs. To realize video-rate 3D shape reconstruction, we have to develop sophisticated video capturing hardwares.

4. DEFORMABLE 3D MESH MODEL FOR ACCURATE 3D SHAPE RECONSTRUCTION

As is well known, the volume intersection method cannot reconstruct accurate 3D object shape. That is, its output represents just the visual hull of the object; concave portions of the object cannot be reconstructed. To overcome this problem and reconstruct accurate 3D object shape, we developed a deformable mesh model based algorithm.

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

With the deformable mesh model, we can employ geometric and photometric constraints of the object surface into its shape reconstruction process, which are not used in the volume intersection method, stereo, or the space carving method[11].
Our algorithm consists of the following steps:

step 1 Convert the voxel representation into the triangle mesh model by the discrete marching cubes algorithm[8] and use it as an initial shape.

step 2 Deform the model iteratively:

  step 2.1 Compute force working at each vertex respectively.

  step 2.2 Move each vertex according to the force.

  step 2.3 Terminate if the vertex motions are small enough. Otherwise go back to 2.1 .

To realize the shape deformation like SNAKES[12], we can use either energy function based or force based methods. As described above, we employed a force based method. This is firstly, from a computational point of view, because we have too many vertices to solve energy function and secondly, from an analytical point of view, because one of the constraints used to control the deformation cannot be represented as any analytical energy function (see below).
We employed the following three types of constraints to control the 3D shape deformation:

1. **Photometric constraint**: a patch in the mesh model should be placed so that its texture, which is computed by projecting the patch onto a captured image, should be consistent irrespectively of onto which image it is projected.

2. **Silhouette constraint**: when the mesh model is projected onto an image plane, its 2D silhouette should be coincide with the observed object silhouette on that image plane.

3. **Smoothness constraint**: the 3D mesh should be locally smooth and should not intersect with itself.

These constraints define a *frame and skin model* to represent 3D object shape:

- Suppose we want to model the object in Figure 8 (a).

- First, the silhouette constraint defines a set of frames of the object (Figure 8 (b)).

- Then the smoothness constraint defines a rubber sheet skin to cover the frames (Figure 8 (c)).

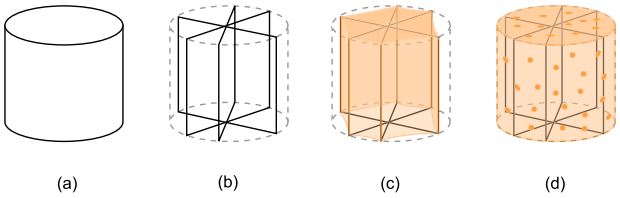


Figure 8: Frame and skin model

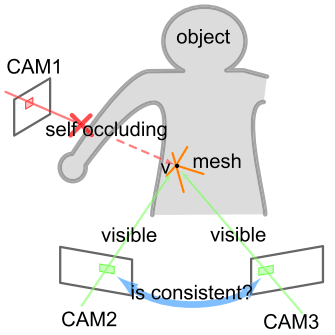


Figure 9: Photometric consistency and visibility

- Finally, the photometric constraint defines supporting points on the skin that have prominent textures (Figure 8 (d)).

In what follows, we describe the forces at each vertex generated to satisfy the constraints.

4.1. Forces at each Vertex

We denote a vertex, its 3D position, and the set of cameras which can observe that vertex by v, q_v, and C_v respectively. For example, $C_v = \{\mathsf{CAM}_2, \mathsf{CAM}_3\}$ in Figure 9.

We introduce the following three forces at v to move its position so that the above mentioned three constraints should be satisfied:

External Force: $F_e(v)$

First, we define external force $F_e(v)$ to deform the mesh to satisfy the photometric constraint.

$$F_e(v) \equiv \nabla E_e(v), \tag{1}$$

where $E_e(v)$ denotes the correlation of textures to be mapped around v (Figure 9) :

$$E_e(v) \equiv \frac{1}{N(C_v)} \sum_{c \in C_v} \left\| p_{v,c} - \overline{p_v} \right\|^2, \tag{2}$$

where c denotes a camera in C_v, $N(C_v)$ the number of cameras in C_v, $p_{v,c}$ the texture corresponding to v on the image captured by c, and $\overline{p_v}$ the average of $p_{v,c}$. F_e moves v so that its corresponding image textures observed by the cameras in C_v become mutually consistent.

Internal Force: $F_i(v)$

Since $F_e(v)$ may destroy smoothness of the mesh or incur self-intersection, we introduce the internal force F_i at v:

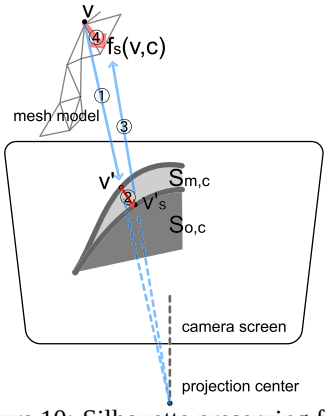


Figure 10: Silhouette preserving force

$$F_i(v) \equiv \frac{\sum_j^n q_{v_j} - q_v}{n}, \tag{3}$$

where q_{v_j} denotes neighboring vertices of v and n the number of such vertices. F_i works like tension between vertices and keeps them locally smooth.

Silhouette Preserving Force: $F_s(v)$

To satisfy the silhouette constraint described before, we introduce the silhouette preserving force $F_s(v)$. This is the most distinguishing characteristics of our deformable model and involves nonlinear selective operation based on the global shape of the mesh, which cannot be analytically represented by any energy function.

Figure 10 explains how this force at v is computed, where $S_{o,c}$ denotes the object silhouette observed by camera c, $S_{m,c}$ the 2D projection of the 3D mesh on the image plane of camera c, and v' the 2D projection of v on the image plane of camera c.

1. For each c in C_v, compute the partial silhouette preserving force $f_s(v,c)$ by the following method.
2. If
   (a) v' is located out of $S_{o,c}$ or
   (b) v' is located in $S_{o,c}$ and on the contour of $S_{m,c}$,

   then compute the 2D shortest vector from v' to $S_{o,c}$ (Figure 10 ②) and set its corresponding 3D vector = $f_s(v,c)$ (Figure 10 ④).
3. Otherwise, $f_s(v,c) = 0$.

The overall silhouette preserving force at v is computed by summing up $f_s(v,c)$:

$$F_s(v) \equiv \sum_{c \in C_v} f_s(v,c). \tag{4}$$

Note that $F_s(v)$ works only at those vertices that are located around the object contour generator[13], which is

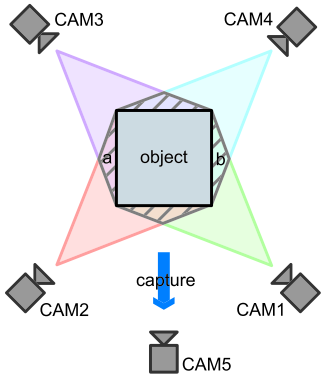
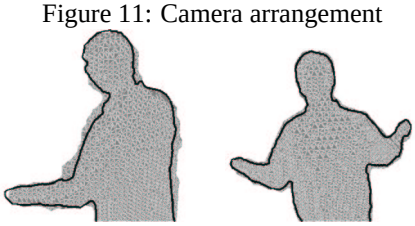


Figure 11: Camera arrangement



(a) CAM5 (b) CAM1

Figure 12: Initial shape. (a) is viewed from CAM_5 in Figure 11, (b) from CAM_1

defined based on the global 3D shape of the object as well as locations of image planes of the cameras.

Overall Vertex Force: $F(v)$

Finally we define vertex force $F(v)$ with coefficients α, β, γ as follows:

$$F(v) \equiv \alpha F_i(v) + \beta F_e(v) + \gamma F_s(v). \quad (5)$$

$F_e(v)$ and $F_s(v)$ work to reconstruct the accurate object shape and $F_i(v)$ to smooth and interpolate the shape. Note that there may be some vertices where $C_v = \{\}$ and hence $F_e(v) = F_s(v) = 0$.

4.2. Performance Evaluation

Figure 11 illustrates the camera arrangement for experiments, where we use $\mathsf{CAM}_1, \ldots, \mathsf{CAM}_4$ for the shape reconstruction and CAM_5 for the performance evaluation. That is, we compare the 2D silhouette of the reconstructed shape viewed from CAM_5 position with the really observed one by CAM_5.

Figure 12 shows the initial object shape by the volume intersection using the images captured by $\mathsf{CAM}_1, \ldots, \mathsf{CAM}_4$. The shape is viewed from CAM_5 and CAM_1 positions, i.e. the shaded regions show $S_{m,5}$ and $S_{m,1}$, respectively. Bold lines in the figures highlight the contours of $S_{o,5}$ and $S_{o,1}$. We can observe some differences between $S_{o,5}$ and $S_{m,5}$ while not between $S_{o,1}$ and $S_{m,1}$. This is because the image captured by CAM_5 is used for the reconstruction.

In the experiments, we evaluated our algorithm with the following conditions : (a) $F(v) = F_i(v)$, (b) $F(v) = F_i(v) + F_p(v)$, (c) $F(v) = F_e(v) + F_i(v) + F_p(v)$. The first row of Figure 13 illustrates $S_{m,5}$ (left) and $S_{m,1}$ (right) for each condition associated with the bold lines denoting the corresponding observed object silhouette contours: $S_{o,5}$ and $S_{o,1}$. The graphs on the second row show how the average error between $S_{m,c}$ and $S_{o,c}$ $c = 1, 5$ changes with the iterative shape deformation.

From these results we can get the following observations:

- With $F_i(v)$ alone (Figure 13(a)), the mesh model shrinks and its 2D silhouette on each image plane becomes far apart from the observed one.
- With $F_i(v)$ and $F_s(v)$, while $S_{m,c}, c = \{1 \ldots 4\}$ well match with $S_{m,c}$, $S_{m,5}$, whose corresponding image is not used for the reconstruction, does not deform well (Figure 13(b)).
- With $F_e(v)$, $F_i(v)$, and $F_s(v)$, $S_{m,5}$ matches well with $S_{o,5}$ (Figure 13(c)). This proves the effectiveness of $F_e(v)$.

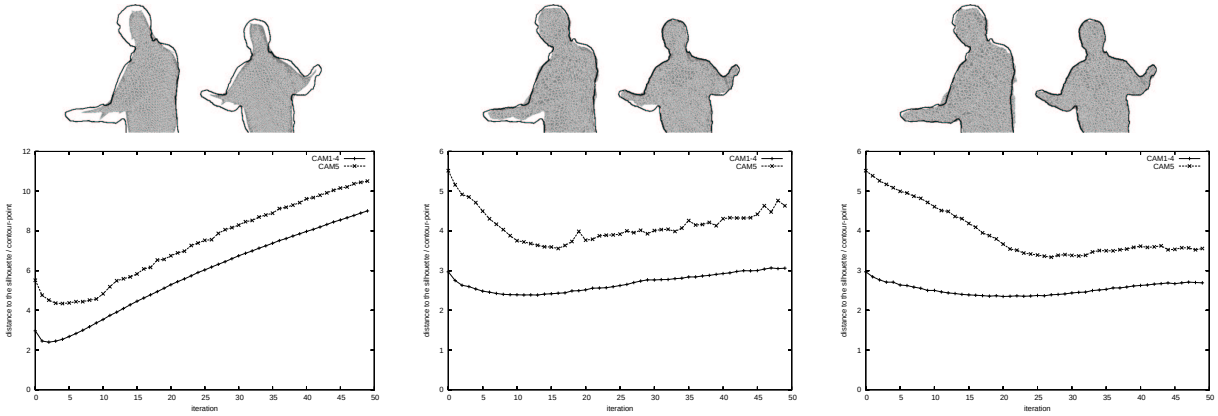
Compared with the Space-Carving method[11], which employs photometric consistency as its main reconstruction cue, our approach additionally employs geometric continuity and silhouette constraint. Such rich constraints make our approach more stable and accurate. Moreover, our deformable mesh model can be extended to dynamic inter-frame deformation, which will enable us to analyze dynamic object motion and realize highly efficient data compression.

5. HIGH FIDELITY TEXTURE MAPPING ALGORITHM

5.1. Viewpoint Dependent Vertex-Based Texture Mapping Algorithm

For 3D video visualization, we developed a texture mapping algorithm that can generate high fidelity object images based on not so accurate 3D shape; the mesh model obtained may not be so accurate because of errors of camera calibration or limitations of our shape reconstruction algorithm.

We first implemented a naive texture mapping algorithm, which selects the most ”appropriate” camera for a patch and then maps onto the patch the texture extracted from the image observed by the selected camera. The appropriateness is measured based on the normal vector of a patch and the viewing direction of a camera. Since this texture mapping is conducted independently of the viewer’s viewpoint of 3D video, we call it as the Viewpoint Independent Patch-Based Method (VIPBM in short).



(a) (b) (c)

Figure 13: Experimental results. (a)$F_i(v)$ alone ($\alpha = 1.0, \beta = 0.0, \gamma = 0.0$), (b)$F_i(v) + F_p(v)$ ($\alpha = 0.5, \beta = 0.0, \gamma = 0.5$) (c)$F_i(v) + F_e(v) + F_s(v)$ ($\alpha = 0.3, \beta = 0.4, \gamma = 0.3$)

From the viewpoint of fidelity, however, the rendered image quality is not satisfiable;

1. Due to errors involved in the patch normal estimation, the best camera for a patch varies from patch to patch even if patches are neighboring. Thus, textures on neighboring patches are often extracted from those images captured by different cameras, which introduces jitters in rendered images.

2. Since the texture mapping is conducted patch by patch and their positions are not accurate, textures of neighboring patches may not be smoothly connected even if their textures are taken from the same observed image. This introduces jitters at patch boundaries in rendered images.

To overcome these quality problems, we developed a viewpoint dependent vertex-based texture mapping algorithm[7]. In this algorithm, the color (i.e. RGB values) of a vertex is computed taking into account of the viewpoint of a viewer and then the texture of a patch is generated by interpolating color values of its three vertices.
Our algorithm can be summarized as follows:

1. Specify the viewline vector. It denotes the unit normal vector of the image plane on which the object image is to be rendered.

2. Then compute inner product between the viewline vector and that of each camera. We use the computed value as the weighting factor for each camera.

3. For each vertex, compute its RGB values based on both those RGB values of the pixels in the observed images corresponding to the vertex and the weighting factors of the cameras.

4. For each triangle, generate texture by linearly interpolating the RGB values of its 3 vertices.

5.2. Performance Evaluation

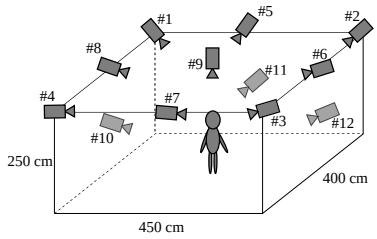


Figure 14: Camera Setting

To evaluate the performance of the proposed viewpoint dependent vertex-based method (VDVBM), we compare it with the viewpoint independent patch-based method (VIPBM). To quantitatively evaluate the quality, we calculate RGB root-mean-square (rms) errors between a real image captured by a camera c and its corresponding images generated by VIPBM and VDVBM, respectively. To evaluate the performance of VDVBM, we employed two methods: VDVBM–1 generates images based on a set of observed images including one captured by camera c itself, while VDVBM–2 excludes such real image captured by camera c. The experiments were conducted under the following settings:

- camera configuration: Figure 14
- image size: 640×480[pixel] 24 bit RGB color
- viewpoint c: camera 5 in Figure 14

Figure 15 illustrates rms errors from frame 95 to 145. This figure proves that VDVBMs performs better than VIPBM. The superiority of VDVBM and its high fidelity image generation capability can be easily observed in Figure 16, where real and generated images for frame 110 and 120 are illustrated.

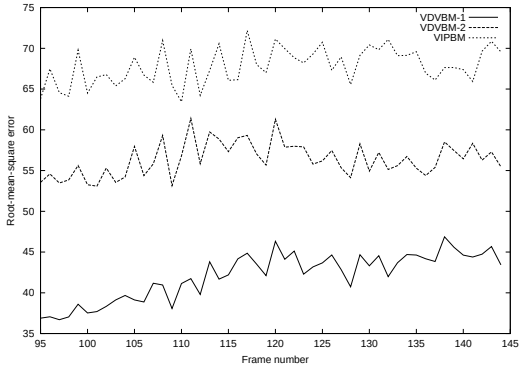


Figure 15: Root-mean-square errors of RGB values (top: VIPBM, middle: VDVBM-2, bottom: VDVBM-1).

Finally, Figure 17 show examples of edited 3D video, where a dancing lady is copied and placed in different 3D locations and an omnidirectional background image is introduced.

6. CONCLUSION

We are proposing 3D video as new image media: it records the object's full 3D shape, motion, and surface properties (i.e. color and texture). In this paper, we presented research attainments so far obtained:

1. a PC cluster system for real-time reconstruction of dynamic 3D object actions from multi-viewpoint video images,
2. a deformable 3D mesh model for reconstructing the accurate 3D object shape, and
3. an algorithm of rendering high fidelity texture on the 3D object surface from the multi-viewpoint video images.

To make 3D video usable in everyday life, we still have to develop methods of

- effective data compression
- more natural image generation
- higher speed and more accurate 3D action reconstruction
- editing 3D video for artistic image contents.

Based on these novel technologies, we will be able to open up new image media world and promote personal and social activities in education, culture, entertainment, sport, and so on.

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

7. REFERENCES

[1] S. Moezzi, L. Tai, and P. Gerard, "Virtual view generation for 3d digital video," *IEEE Multimedia*, pp. 18–26, 1997.

[2] T. Kanade, P. Rander, and P. J. Narayanan, "Virtualized reality: Constructing virtual worlds from real scenes," *IEEE Multimedia*, pp. 34–47, 1997.

[3] T. Wada, X. Wu, S. Tokai, and T. Matsuyama, "Homography based parallel volume intersection: Toward real-time reconstruction using active camera," in *Proc. of International Workshop on Computer Architectures for Machine Perception*, Padova, Italy, Sept. 2000, pp. 331–339.

[4] E. Borovikov and L. Davis, "A distributed system for real-time volume reconstruction," in *Proc. of International Workshop on Computer Architectures for Machine Perception*, Padova, Italy, Sept. 2000, pp. 183–189.

[5] G. Cheung and T. Kanade, "A real time system for robust 3d voxel reconstruction of human motions," in *Proc. of Computer Vision and Pattern Recognition*, South Carolina, USA, June 2000, pp. 714–720.

[6] T. Matsuyama and R. Yamashita, "Requirements for standardization of 3d video," *ISO/IEC JTC1/SC29/WG11, MPEG2002/M8107*, 2002.

[7] T. Matsuyama and T. Takai, "Generation, visualization, and editing of 3d video," in *Proc. of symposium on 3D Data Processing Visualization and Transmission*, Padova, Italy, June 2002, pp. 234–245.

[8] Y. Kenmochi, K. Kotani, and A. Imiya, "Marching cubes method with connectivity," in *Proc. of 1999 International Conference on Image Processing*, Kobe, Japan, Oct. 1999, pp. 361–365.

[9] H. Tezuka, A. Hori, Y. Ishikawa, and M. Sato, "Pm: An operating system coordinated high performance communication library, *high-performance computing and networking*, lecture notes in computer science, vol. 1225," 1997.

[10] T. Wada and T. Matsuyama, "Appearance sphere : Background model for pan-tilt-zoom camera," in *Proc. of 13th International Conference on Pattern Recognition*, Wienna, Austria, Aug. 1996, pp. A–718–A–722.

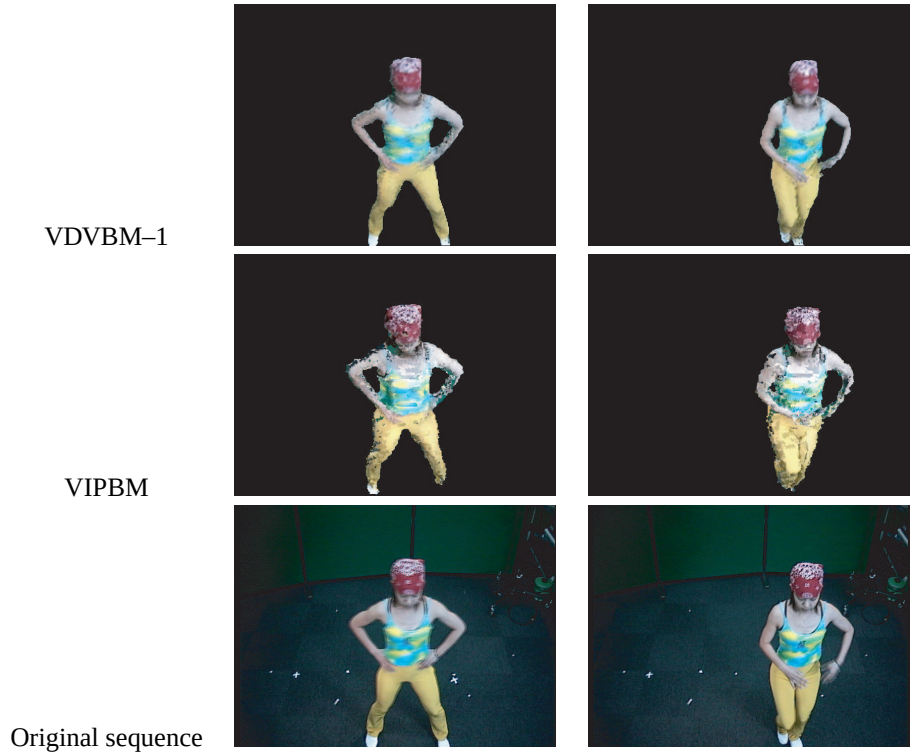


Figure 16: Sample frames of generated 3D video

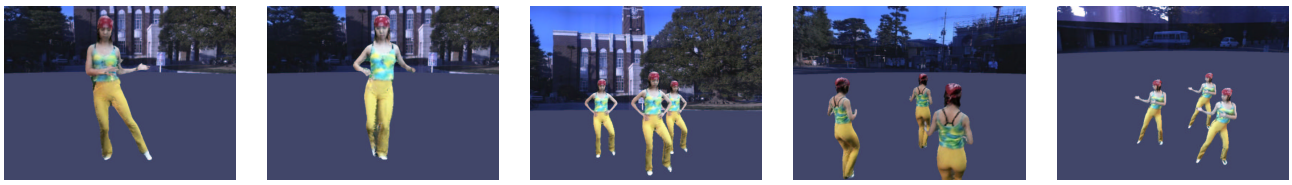


Figure 17: 3D video with an omni-directional background

[11] K. N. Kutulakos and S. M. Seitz, "A theory of shape by space carving," in *Proc. of International Conference on Computer Vision*, Kerkyra, Greece, Sept. 1999, pp. 307–314.

[12] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331, 1988.

[13] Geoffrey Cross and Andrew Zisserman, "Surface reconstruction from multiple views using apparent contours and surface texture," 2000.