

# Exploitation of 3D Video Technologies

Takashi Matsuyama  
Graduate School of Informatics  
Kyoto University  
Sakyo, Kyoto 606-8501, Japan  
tm@i.kyoto-u.ac.jp

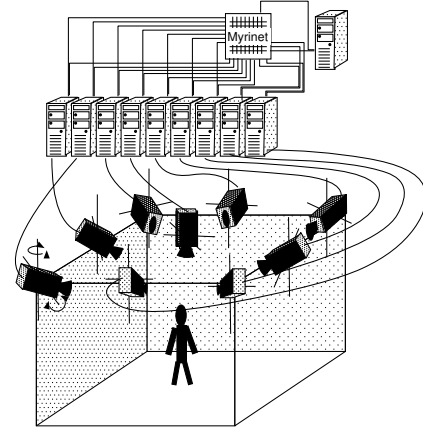
## Abstract

*3D video is NOT an artificial CG animation but a real 3D movie recording the full 3D shape, motion, and precise surface color & texture of real world objects. It enables us to observe real object behaviors from any viewpoints as well as to see pop-up 3D object images. We believe the exploitation of 3D video technologies will open up a new world of versatile cultural and social activities: visual communication, entertainment, training & learning, archiving, and so on. This paper gives an overview of our research attainments so far obtained: (1) a PC cluster system with distributed active cameras for real-time 3D shape reconstruction, (2) a dynamic 3D mesh deformation method for obtaining accurate 3D object shape, (3) a texture mapping algorithm for high fidelity visualization, and (4) user friendly 3D video editing system.*

## 1. Introduction

3D video[5] is NOT an artificial CG animation but a real 3D movie recording the full 3D shape, motion, and precise surface color & texture of real world objects. It enables us to observe real object behaviors from any viewpoints as well as to see pop-up 3D object images. Such new featured image medium will promote wide varieties of personal and social human activities: communication (e.g. 3D TV phone), 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.

This paper gives an overview of our research attainments so far obtained: (1) a PC cluster system with distributed active cameras for real-time 3D shape reconstruction, (2) a dynamic 3D mesh deformation method for obtaining accurate 3D object shape, (3) a texture mapping algorithm for high



**Figure 1. PC cluster for real-time active 3D object shape reconstruction.**

fidelity visualization, and (4) user friendly 3D video editing system<sup>1</sup>.

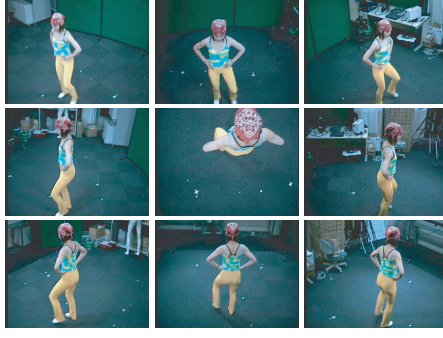
## 2. Real-Time 3D Shape Reconstruction

### 2.1. System Organization

Figure 1 illustrates the architecture 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), which enables us to implement efficient parallel processing on the PC cluster.
- Distributed active video cameras: Among 30, 25 PCs have Fixed-Viewpoint Pan-Tilt (FV-PT) cameras[6], respectively, for active object tracking and imaging.

<sup>1</sup> As for technical details, please refer to [7][2][4][3]. Figures and parts of the text are cited from these papers.



**Figure 2. Captured multi-viewpoint images**

Figure 2 shows a snapshot of multi-view object video data captured by the system.

## 2.2. Basic Scheme of 3D Video Generation

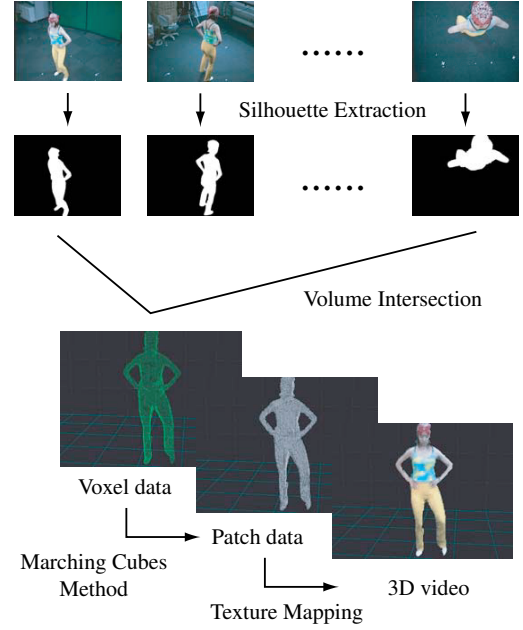
Figure 3 illustrates the basic process of generating a 3D video frame in our system:

1. **Synchronized Multi-View Image Acquisition:** A set of multi-view object images are taken simultaneously (top row in Figure 3).
2. **Silhouette Extraction:** Background subtraction is applied to each captured image to generate a set of multi-view object silhouettes (second top row in Figure 3).
3. **Silhouette Volume Intersection:** As shown in Figure 4, 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 3).
4. **Surface Shape Computation:** The discrete marching cubes method[1] is applied to convert the voxel representation to the surface mesh representation. Then the generated 3D mesh is deformed to obtain accurate 3D object shape (second bottom in Figure 3).
5. **Texture Mapping:** Color and texture on each patch are computed from the observed multi-view images (bottom in Figure 3).

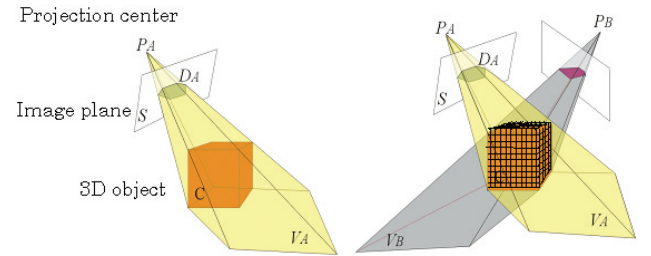
By repeating the above process for each video frame, we have a live 3D motion picture.

## 2.3. Parallel Volume Intersection Algorithm Using Plane-to-Plane Perspective Projection

The back-projection is the most expensive computation in the above volume intersection method. To accelerate the computation, we first developed the plane-to-plane perspective projection (PPPP) algorithm, where the 3D voxel space



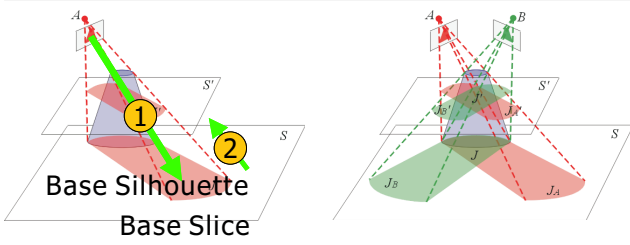
**Figure 3. 3D video generation process**



**Figure 4. Visual cone intersection for 3D shape reconstruction**

is partitioned into a group of parallel planes and the cross-section of the 3D object volume on each plane is reconstructed (Figure 5):

1. First an object silhouette in the image plane is back-projected on the base plane in the 3D space.
2. Then the back-projected base silhouette is projected onto each of the parallel planes.
3. Finally, back-projected silhouettes on each plane are intersected with each other to generate an object cross-section on that plane. By stacking up such cross-sections, we have the voxel representation of the 3D object shape.



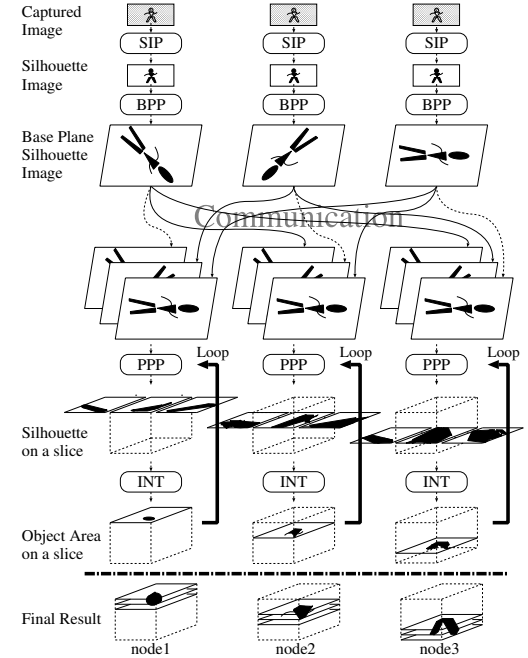
**Figure 5. Plane-to-plane perspective projection algorithm**

The next step to realize real-time 3D shape reconstruction is to introduce parallel processing by the PC cluster. Figure 6 illustrates the processing flow of the parallel pipelined PPPP algorithm:

1. **Image Capture** : Triggered by a capturing command, each PC with a camera captures a video frame (Figure 6 top row).
2. **Silhouette Extraction** : Each PC extracts an object silhouette from the video frame (Figure 6 second top row).
3. **Projection to the Base-Plane** Each PC projects the silhouette onto the common base-plane in the 3D space (Figure 6 third top row).
4. **Base-Plane Silhouette Duplication** : All base-plane silhouettes are duplicated across all PCs over the network so that each PC has the full set of base-plane silhouettes (Figure 6 forth top row).
5. **Object Cross Section Computation** : Each PC computes object cross sections on specified parallel planes in parallel (Figure 6 three bottom rows).

The above processing is implemented as the 5 stage pipeline process. The experimental results showed that the PC cluster system can reconstruct a full 3D shape of a human in 80 ~ 90ms, i.e. 11 ~ 12 frame per second under such conditions that the size of input image is  $640 \times 480$  pixels, the voxel size is  $2\text{cm} \times 2\text{cm} \times 2\text{cm}$ , and the 3D shape is contained in a space of  $2\text{m} \times 2\text{m} \times 2\text{m}$ .

In summary, the experiments proved the effectiveness of the proposed real-time 3D shape reconstruction system: the plane-based volume intersection method and the parallel pipeline implementation. Moreover, the proposed parallel processing method is flexible enough to scale up the system by increasing numbers of cameras and PCs.



**Figure 6. Processing flow of the parallel pipelined 3D shape reconstruction.**

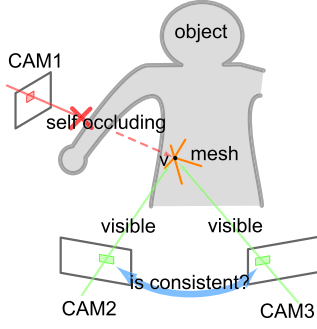
### 3. Deformable Mesh Model for Accurate 3D Shape Reconstruction

As is well known, the volume intersection method is stable, but cannot reconstruct accurate 3D object shape; its output represents just the visual hull of the object and concave portions of the object cannot be reconstructed.

To solve this problem, we proposed a deformable mesh model: we first convert the reconstructed 3D voxel data into a surface mesh consisting of triangular surface patches and then deform it to fit the object surface.

The deformation is conducted to satisfy the following constraints:

- 1) **photo-consistency constraint** : when a patch is mapped onto captured multi-view images, colors and textures of the mapped patches should be mutually consistent (Figure 7).
- 2) **silhouette constraint** : when the mesh is mapped onto each captured image, the silhouette of the mapped mesh should be aligned with the observed object silhouette.
- 3) **smoothness constraint** : the mesh should be smooth and should not cause any self intersection.
- 4) **3D motion flow constraint** : the mesh should be dynamically deformed according to object actions.



**Figure 7. Photometric consistency**

**5) inertia constraint** : the dynamic mesh deformation should be smooth.

In our 3D deformable mesh model, we introduce two types of deformation: intra-frame deformation and inter-frame deformation. In the intra-frame deformation, our model uses the visual hull, a result of the volume intersection, as the initial shape and changes its shape so as to satisfy constraints 1), 2) and 3) described above. While the volume intersection employs the geometric information (i.e., silhouettes) alone, the deformable model refines the 3D shape using photometric information. As a result, we can reconstruct the accurate 3D object shape even for its concave portions.

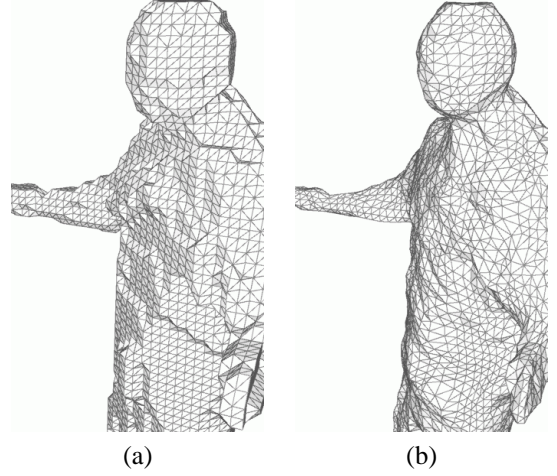
In the inter-frame deformation, on the other hand, the mesh changes its shape frame by frame to satisfy all constraints 1), ..., 5). This deformation process enables us to obtain a temporal sequence of topologically consistent 3D meshes. That is, since each vertex of the mesh can be traced over time, we can easily analyze detailed object motion.

### 3.1. Intra-Frame Deformation

Our intra-frame deformation algorithm consists of the following steps:

- step 1** Convert the voxel representation into a triangle mesh model by the discrete marching cubes algorithm[1], which is used as the initial shape for the deformation.
- step 2** Deform the mesh iteratively:
  - step 2.1** Compute force acting on each vertex: the force denotes how much energy is required to satisfy the constraints 1), 2), and 3).
  - step 2.2** Move each vertex according to the force.
  - step 2.3** Terminate if all vertex motions are small enough. Otherwise go back to 2.1 .

Figure 8(a) illustrates a close-up of the mesh data generated from reconstructed voxel data. Figure 8(b) illustrates



**Figure 8. (a) surface mesh generated by the discrete Marching cube method and (b) surface mesh after the intra-frame deformation**

the result of the intra-frame deformation, where accurate and smooth 3D object shape is obtained.

### 3.2. Inter-Frame Deformation


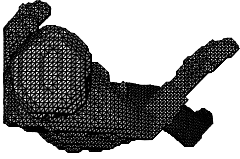
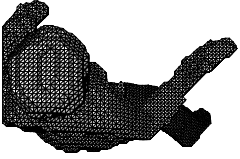

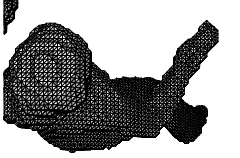
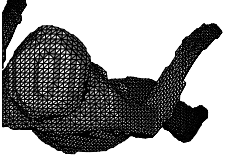

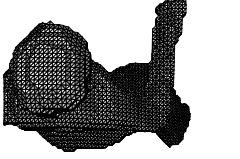
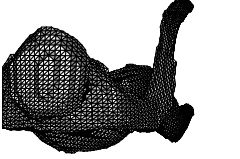
If a mesh at time  $t$  deforms its shape to satisfy the constraints at time  $t+1$ , we can obtain the shape at  $t+1$  and the motion from  $t$  to  $t+1$  simultaneously. Note that by this dynamic deformation, the topological structure of the mesh is preserved and hence we can compute the motion vector for each vertex.

To realize this dynamic mesh deformation, i.e. the inter-frame deformation, we introduce the 3D motion flow and inertia constraints in addition to the photo-consistency, silhouette, and smoothness constraints described before.

Figure 9 and 10 illustrate the inter-frame deformation through 3 successive frames. The columns of Figure 9 show, from left to right, the captured images, the visual hulls generated by the discrete marching cubes method, and the mesh data generated by the inter-frame deformation, respectively. Note that the visual hull in frame  $t$  was used as the initial shape for the intra-frame deformation and then the resultant mesh for the inter-frame deformation.

From these results, we can observe:

- Our dynamic mesh model can follow the non-rigid object motion smoothly.
- During its dynamic deformation, our mesh model preserves both global and local topological structure and hence we can find corresponding vertices between any pair of frames for all vertices. Figure 10 illustrates this topology preserving characteristic. That is, the left

Frame	Captured image	Visual hull	Deformable mesh model
$t$			
$t + 1$			
$t + 2$			

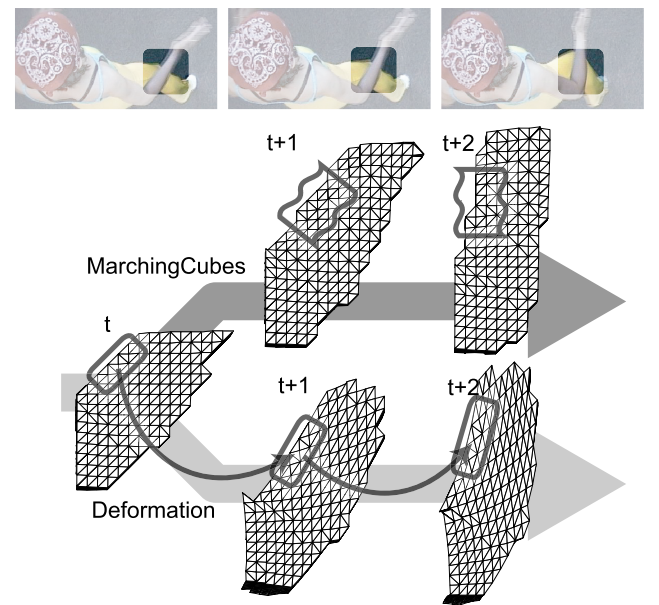
**Figure 9. Results of the inter-frame deformation (overview)**

mesh denotes a part of the initial mesh obtained by applying the marching cubes to the visual hull at  $t$ . The lower bold arrow stands for the inter-frame deformation process, where any parts of the mesh can be traced over time. Aligned along the upper bold arrow, on the other hand, are parts of the meshes obtained by applying the marching cubes to each visual hull independently, where no vertex correspondence can be established because the topological structures of the meshes are different.

In summary, we can demonstrate the effectiveness of our dynamic deformable mesh model for obtaining accurate dynamic 3D object actions. The mesh data we used consist of about 12,000 vertices and 24,000 triangles, and the processing time per frame is about 10 minutes by PC (Xeon 1.7GHz); we do not study any real-time processing for the deformation process yet.

#### 4. High Fidelity Texture Mapping Algorithm

In this section, we propose two texture mapping algorithms to generate high fidelity 3D video and compare their performance.



**Figure 10. Results of the inter-frame deformation (close-up)**



## 4.1. Texture Mapping Algorithms

We first implemented a naive texture mapping algorithm, which selects the most "appropriate" camera for each patch and then maps onto the patch the texture extracted from the image observed by the selected 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).

This method generates fully textured 3D object shape, which can be viewed from arbitrary viewpoints with ordinary 3D graphic display systems. Moreover, its data size is very compact compared with that of the original multi-viewpoint video data.

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

1. Due to the rough quantization of patch normals, the best camera for a patch varies from patch to patch even if they are neighboring. Thus, textures on neighboring patches are often extracted from those images captured by different cameras (i.e. viewpoints), which introduces jitters in displayed images.
2. Since the texture mapping is conducted patch by patch and their normals are not accurate, textures of neighboring patches may not be smoothly connected. This introduces jitters at patch boundaries in displayed images.

To overcome these quality problems, we developed a viewpoint dependent vertex-based texture mapping algorithm (VDVBM in short). In this algorithm, the color (i.e. RGB values) of each vertex of patches is computed taking into account of the viewpoint of a viewer and then the texture of each patch is generated by interpolating color values of its three vertices.

## 4.2. Performance Evaluation

To evaluate the performance of VDVBM and the effectiveness of the mesh deformation, we applied VIPBM and VDVBM to **Mesh** (converted from voxel data) and **D-Mesh** (after deformation) respectively and evaluated the generated images qualitatively.

Figures 11 and 12 show images generated by VIPBM and VDVBM, respectively, using the same frame data of **Mesh** and **D-Mesh** and the same viewpoints. From these images, we can observe

- Comparing those images generated by VIPBM and VDVBM, the former introduces many jitters in images, which are considerably reduced by the latter.
- Comparing those images generated with **Mesh** and **D-Mesh**,



Figure 11. Images generated by the Viewpoint Independent Patch-Based Method



Figure 12. Images generated by the Viewpoint Dependent Vertex-Based Method

- VIPBM with **D-Mesh** can generate better images; while many jitters are still included, detailed textures can be observed. This is because the surface normals are smoothed and become more accurate by the mesh deformation.
- On the other hand, VDVBM with **D-Mesh** does not show any observable improvements and instead seems to introduce more blurring effects. This is because in VDVBM, the viewpoint information to generate an image plays much more important role than the accuracy of the 3D shape. In other words, VDVBM can work well even for 3D object shape data of limited accuracy.

Figure 13 compares images generated by VIPBM and VDVBM with **D-Mesh** to their corresponding original video images. This also verify the effectiveness of VDVBM.

Next, we conducted quantitative performance evaluations of VIPBM and VDVBM with **Mesh** and **D-Mesh**. We calculate RGB root-mean-square (rms) errors between a real image captured by camera  $c_j$  and its corresponding images generated by VIPBM and VDVBM, respectively: in generating the images, the position and direction of camera  $c_j$  are used as those of the viewpoint for the 3D video. To evaluate the performance of VDVBM, we employed two methods: VDVBM-1 generates images by using all multi-view images including real images captured by camera  $c_j$  itself, while VDVBM-2 excludes such real images captured by camera  $c_j$ . The experiments were conducted under the

VDVBM



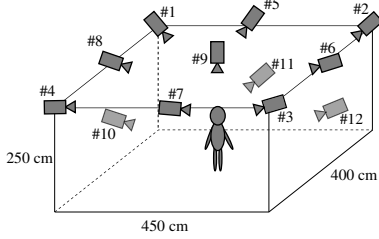
VIPBM

Original  
sequence

frame # 106

frame # 126

**Figure 13. Sample images of generated 3D video with D-Mesh**



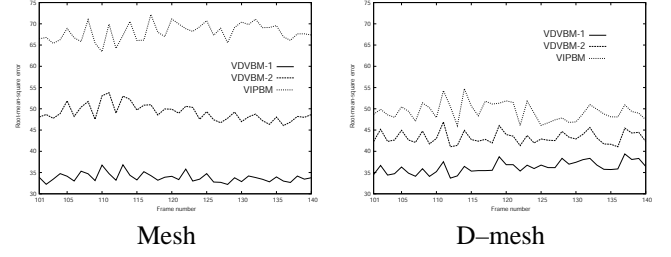
**Figure 14. Camera Setting**

following settings:

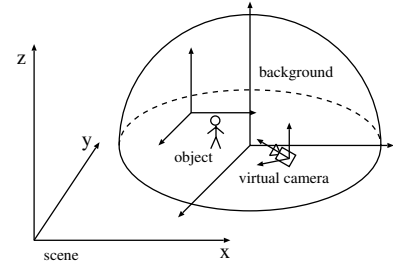
- camera configuration: Figure 14
- image size:  $640 \times 480$ [pixel] 24 bit RGB color
- viewpoint: camera 5

Figure 15 illustrates the experimental results, where rms errors for frame 101 to 140 are computed. This figure proves that

- VDVBM-1 and VDVBM-2 perform better than VIPBM with both **Mesh** and **D-Mesh**.
- Comparing **Mesh** with **D-Mesh**,
  - In VIPBM, **D-Mesh** improves the fidelity of generated images significantly.



**Figure 15. Root-mean-square errors of RGB values**



**Figure 16. Virtual Scene Setup**

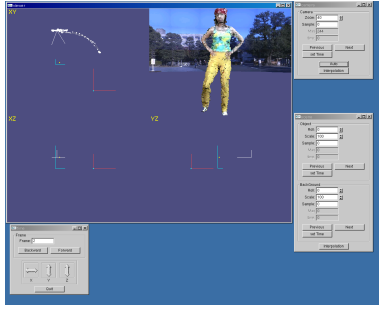
- In VDVBM-2, **D-Mesh** reduces rms errors about 10%.
- In VDVBM-1, on the other hand, **D-Mesh** increases rms errors slightly. This is because **D-Mesh** introduces blurring effects as discussed before.

In summary, the proposed texture mapping method (VDVBM) works well to render natural looking object images from any viewpoint. Moreover, the the mesh deformation is effective for 3D video visualization as well as for accurate 3D shape reconstruction.

## 5. Editing and Visualization System of 3D Video

To visualize 3D video, we first define a virtual scene consisting of a reconstructed 3D object, a background and a virtual camera. Figure 16 illustrates the virtual scene we used, where a dome-shaped omnidirectional background image is introduced as a virtual background. This image is generated by mosaicing multiple images taken by our FV-PT[6] camera. The background enables us to discriminate camera actions from object motions.

To visualize the scene, we have to specify many parameters including



**Figure 17. a GUI for 3D video editing and visualization.**

- virtual camera parameters: viewpoint, view direction, angle of view, focal length
- object position, pose, and scale
- background position, pose, and scale

Furthermore, these parameters must be specified at each frame, and they should be continuously changed for smooth visualization and natural camera-works.

We developed two methods to help the visualization:

1. **Key Frame Method:** generating camera-works by temporal interpolation of parameters specified for arbitrary key frames.
2. **Automatic Camera-Work Generation Method:** generating camera-works by utilizing an object's parameters (e.g. height, direction).

We also developed a GUI to interactively specify the parameters (Figure 17). In this interface, a user can specify the positions, rotations, and scales of the three coordinate systems: the object coordinate system, the background coordinate system, and the virtual camera's. The top, front, and side views of these coordinate systems and the obtained image are displayed in the left top of the window (Figure 17). Figure 18 illustrates sample shots generated by using the 3D video editing system, where we can copy a 3D video object and arrange its copies as we like.

## 6. Conclusion

In this paper, we presented research attainments so far obtained to generate 3D video: 1. A PC cluster system for real-time reconstruction of 3D object actions from multi-view video images. 2. A deformable 3D mesh model for reconstructing the accurate 3D object shape and motion. 3. An algorithm of rendering high fidelity texture on the reconstructed 3D object surface from the multi-view video images. 4. A user-friendly editing and visualization system.



**Figure 18. Visualized 3D video with an omnidirectional background**

While we believe that the proposed methods set a milestone to realize 3D video, we still have to augment them to make 3D video usable in everyday life.

This work was supported by the grant-in-aid for scientific research (A) 13308017. We thank all members of our laboratory for their helps and insightful suggestions.

## References

- [1] Y. Kenmochi, K. Kotani, and A. Imiya. Marching cubes method with connectivity. In *Proc. of 1999 International Conference on Image Processing*, pages 361–365, Kobe, Japan, Oct. 1999.
- [2] T. Matsuyama and T. Takai. Generation, visualization, and editing of 3d video. In *Proc. of symposium on 3D Data Processing Visualization and Transmission*, pages 234–245, Padova, Italy, June 2002.
- [3] T. Matsuyama, X. Wu, T. Takai, and S. Nobuhara. Real-time 3d shape reconstruction, dynamic 3d mesh deformation, and high fidelity visualization for 3d video. *International Journal on Computer Vision and Image Understanding*, page (in press), 2004.
- [4] T. Matsuyama, X. Wu, T. Takai, and T. Wada. Real-time dynamic 3d object shape reconstruction and high-fidelity texture mapping for 3d video. *IEEE Trans. on Circuit and Systems for Video Technology*, pages 357–369, 2004.
- [5] S. Moezzi, L. Tai, and P. Gerard. Virtual view generation for 3d digital video. *IEEE Multimedia*, pages 18–26, 1997.
- [6] T. Wada and T. Matsuyama. Appearance sphere : Background model for pan-tilt-zoom camera. In *Proc. of 13th International Conference on Pattern Recognition*, pages A-718–A-722, Vienna, Austria, Aug. 1996.
- [7] 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*, pages 331–339, Padova, Italy, Sept. 2000.