# Toward Cinematizing Our Daily Lives

Hansung Kim, Ryuuki Sakamoto, Itaru Kitahara,
Tomoji Toriyama and Kiyoshi Kogure

## Abstract

We introduce a cinematographic video production system to create movie-like attractive footage from our indoor daily life. Since the system is designed for ordinary users in non-studio environments, it is composed of standard hardware components, provides a simple interface, and works in near real-time of 5~6 frames/sec. The proposed system reconstructs a visual hull from acquired multiple videos and then generates final videos from the model by referring to the camera shots used in film-making. The proposed method utilizes "Reliability" to compensate for errors that may have occurred in non-studio environments and to produce the most natural scene from the reconstructed model. By using a virtual camera control system, even non-experts can easily convert the 3D model to movies that look as if they were created by experienced filmmakers.

**Keywords:** 3D video system, Cinematized Reality, Multiple camera system, Cinematographic camera control

# 1. INTRODUCTION

Many studies have already used video cameras to capture and store daily experiences as external memories. Small events and incidents can happen anytime and anywhere in daily life. Capturing such moments would be useful for cherishing memories, making life-logs, or learning from experiences.

There are two typical approaches to capturing and storing daily experiences. The first, which uses a wearable or portable camera, can easily record all activities the object performs with a single video stream. However, in daily life, continuously grasping the important moments from the best viewpoint is difficult with a portable camera. The second approach uses fixed environmental cameras that always provide objective and stable visual information, but an enormous amount of unnecessary video must be captured to cover the entire area at all times. We also have to switch between multiple videos to determine the best viewpoint for observing the activities.

To solve the above problems, studies have been conducted on regenerating video captured at arbitrary view-points in 3D space. These techniques reconstruct 3D models in the space by merging multiple videos and generate free-viewpoint videos by applying computer graphic technology.

Although we can overcome the above 3D reconstruction problems, one more problem remains: how can we produce attractive videos from 3D models? In each scene, there are several choices of shots. Interestingly, different shot choices produce different impressions and effects, even if the captured scene is identical.

Finally, the system must be equipped with a moderate price and show robust performance in real environments with proper processing speed for use in daily life. If the system requires a special environment such as a blue-screen studio or high performance computers, access by average users will be limited. Moreover, if the system cannot provide fast feedback of the results of their data, normal users will give it a “cold shoulder.”

In this paper, we introduce a cinematographic video production system to overcome the above limitations. Its key concept is capturing moments in daily life with multiple cameras and regenerating film-like footage. The system consists of two subsystems: free-viewpoint video generation and cinematographic camera control. The free-viewpoint video system generates fine 3D models using multiple cameras, and the cinematographic camera control system helps users generate final videos from the 3D model by referring to professional camera works.

## 1.1. Related Works

**Shape-from-silhouette.** Since Kanade et al. proposed “Virtualized Reality” as a new visual medium for manipulating and rendering prerecorded scenes [1], many computer vision-based 3D imaging systems have been developed around the shape-from-silhouette (SFS) approach [2][3]. Silhouettes are

easily obtainable with several restrictions and the implementation of the shape-from-silhouette method is generally straightforward. The visual hulls constructed using SFS provide an upper bound on the shape of the object. However, common SFS method has the following limitations for use in real environments.

First, most SFS methods assume perfect silhouette information of objects in advance or use simple background subtraction techniques in restricted environments such as blue screen or simple-colored backgrounds. Grauman et al. proposed a Bayesian approach to compensate for modeling errors from false segmentation [4]. They modeled prior density using probabilistic principal components analysis and estimated maximum-a-posteriori reconstruction of multi-view contours. Guillemaut et al. simultaneously solved both segmentation and reconstruction problems using a Bayesian framework and a graph-cut approach [5].

Second, most previous methods assume that the modeling space is free of occlusion. Therefore, if the model is occluded by any background object from a certain view, conventional methods will produce an incomplete model. Guan et al. pointed out this problem and suggested making occlusion masks by observing the boundaries of a moving object [6].

Third, computational complexity is one of the biggest problems of SFS methods because they have to back-project each 3D point to all silhouette images. Matsuyama et al. and Ueda et al. used PC clusters for real-time modeling and rendering, but doing so drastically increases sys-tem costs [7][8]. Li et al. used hardware acceleration for visual hull rendering [9].

**Virtual view rendering.** One of the most popular methods of rendering voxel-based data is the marching cube algorithm originally proposed by Lorensen and Cline [10]. Lookup tables and vertex interpolation allow the appropriate polygons to be generated for each cube. The resulting mesh is very fine, but it has a very high polygon count and requires high computational load to optimize the mesh structures. The second problem concerns implementation of the algorithm on graphics processing units (GPU). Unfortunately, at present the geometry shader can only take a maximum of six vertices as inputs, so dealing with a full cube at a time is impossible. The alternative is a variation called marching tetrahedra, which splits each cube into five or six tetrahedra to perform a similar algorithm [11]. An additional benefit of the marching tetrahedra algorithm is that it eliminates the ambiguous cases seen in marching cubes. However, since it is also over-tesselated, the optimization problem re-mains.

**Silhouette extraction.** The background subtraction technique is one of the most common approaches for extracting foreground objects from video sequences because it works very quickly and distinguishes semantic object regions from static backgrounds [12][13]. A Gaussian mixture model, which is the most representative approach [14], has been widely incorporated with other methods

[15][16]. However, these approaches require high computational complexity and have trade-offs in the learning rate. Tuzel et al. proposed using 3D multivariate Gaussians to improve computational efficiency [17], and Elgammal et al. used Kernel Density Estimators (KDE) to quickly adapt to background changes [18]. Several advanced approaches using KDE have also been proposed [19][20]. However, these KDE-based approaches consume a lot of memory when updating recent background statistics. Recently, graphcut-based segmentation approaches are being used in offline 3D modeling systems [21][22], but these approaches were not considered in our system because they are too time-consuming for real-time systems.

**Camera works.** The progress of digital cinema technology has made normal users be able to make their own videos and enjoy a small-scale cinema with a high quality at home [23]. However, it is still difficult for normal people to produce attractive videos with changing camera positions or camera works in response to each captured situation like professional cameramen for film and television video productions. The "grammar of film language" describes the rules of filmmaking to produce easily understandable and attractive footage for audiences [24][25]. There have been several trials to apply this grammar to the 3D CG world and verified its effectiveness by video productions [26][27]. However, this grammar has not yet been applied to real events in the real 3D world. In the 3DCINE workshop at CVPR 2006, numerous state-of-the-art techniques for 3D cinematography were presented [28], and Tenmoku et al. recently proposed a pre-visualization system for film-making using 3D modeling and Mixed Reality technique to support real shooting [29]. There have also been trials to produce movie-like videos from reconstructed 3D models [30][31]. However, since they manually applied camera works to their 3D models, they are too time-consuming to produce scenes. More-over, producing video from their own 3D models with such professional 3D editing tools as MAYA and 3DS MAX is difficult for ordinary users.

## 1.2. Contributions

The final goal of our research is to develop a practical 3D video production system that can be used in daily life by average users. Our proposed system was developed to provide a solution to the problems and limitations previously mentioned. The major contributions of the proposed system can be summarized as follows:

• **Reliability-Based Evaluation Criteria.** A "reliability" factor is defined in this system. This reliability organically connects all stages from segmentation to applying camera works to minimize errors and produces the most natural scene from the model.

• **Wide usability in various environments.** The proposed system includes robust silhouette extraction

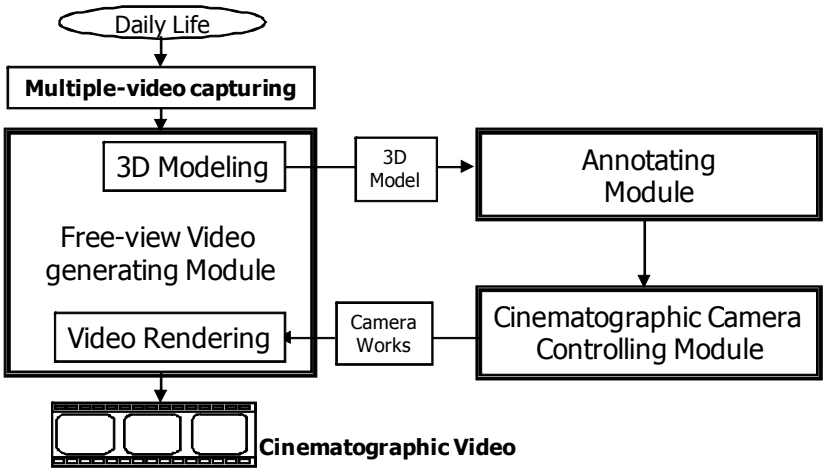



**Fig. 1. Block diagram of cinematographic video production**

and 3D reconstruction algorithms for non-studio environments. Moreover, the system was realized with common PCs and small synchronized IEEE 1394 cameras. Therefore it can be set and shot anyplace with static backgrounds.

• **Interface.** The cinematographic camera control system provides a simple interface to generate movie-like videos by referring to the grammar of film language. Therefore, even non-experts can easily convert their scenes of daily life to attractive movies with the help of the system's expertise.

• **Processing speed.** We realized 3D reconstruction and free-view rendering algorithms on the graphics processing unit (GPU) so that it works in near real-time on a PC. We exploit geometry shaders and shader model 4.0 that were recently added. Therefore, the proposed system provides very fast feedback to users.

## 2. SYSTEM OVERVIEW AND SETUP

The system consists of a multiple-video capturing module, a free-view generation module, an annotation module, and a cinematographic camera-control module as shown in Fig. 1. The capturing module is implemented using seven calibrated IEEE 1394 cameras set on poles or tripods to surround the space. All cameras, which provide 1024x768 RGB video streams at 30 frames/sec, are oriented to observe the capturing space. Each capturing PC extracts the silhouettes of objects and sends silhouette masks and texture information by UDP (User Datagram Protocol) to the modeling PC. The modeling PC features an Intel Core 2 processor running at 2.40-GHz with 2.0-GB of main system memory and a NVIDIA GeForce 8800 GTS video card. The reconstructed 3D model can be directly rendered in

near real-time by controlling a virtual camera with the following methods: keyboard control, ARToolkit [32], or a xml file with previously generated camera works. However, the present system cannot match the capturing speed because segmentation and 3D reconstruction do not work in full real-time of 30 frames/sec. Therefore, the system optionally provides a function to capture and save a scene with time stamps in advance to produce a full frame-rate video on an off-line process. To apply new camera works to the generated model, we need manual processes to annotate the main objects in a scene and select proper cameras shots. With just a few clicks for designating a target object and selecting key frames, camera position, and camera actions, the system automatically produces a final video. The details of each step in the flow are elucidated in the following sections.

# 3. FREE-VIEWPOINT VIDEO GENERATION

The proposed system makes a free-viewpoint video of the real world from multiple cameras. When target objects are captured by these cameras, the PC allocated to each capturing camera segments the objects and transmits the masks and color textures to a 3D modeling server by the system's network. The modeling server then generates 3D models of each object from the gathered masks. Finally, the server generates a video at the designated point of view with the 3D model and texture information.

## 3.1. Silhouette Extraction

Silhouette extraction is one of the most important parts in a SFS-based 3D reconstruction system because segmentation errors directly affect the final model. In our system, the background is modeled using a generalized Gaussian family (GGF) distribution to cope with problems from various changes in the background and shadows [33][34]. Pixel variation in a static scene over time is modeled with the GGF distribution, defined as:

$$p(x:\rho)=\frac{\rho\gamma}{2\Gamma(1/\rho)}\exp\left(-\gamma^{\rho}\,|x-\mu|^{\rho}\right) \quad \text{with} \quad \gamma=\frac{1}{\sigma}\left(\frac{\Gamma(3/\rho)}{\Gamma(1/\rho)}\right)^{1/2} \tag{1}$$

where $\Gamma(\cdot)$ is a gamma function, $\mu$ is a mean, and $\sigma^2$ is a variance of the distribution. In Eq. (1), $\rho$ represents a shape of distribution and it is decided by calculating the excess kurtosis in Eq. (2) of the first *m* frames. For example, the excess kurtosis of Gaussian and Laplace distributions is 0 and 3, respectively.

$$g_2 = \frac{\mu_4}{\sigma^4} - 3 = \frac{m\sum_{i=1}^{m}(x_i - \mu)^4}{\left(\sum_{i=1}^{m}(x_i - \mu)^2\right)^2} - 3 \quad (2)$$

The optimized parameters of the background models can be estimated by maximizing the likelihood of the observed data [35]. The background is modeled in two distinct parts: a luminance component and a hue component. To cope with various changes in the background, the background model is updated by selective running average.

Based on the constructed background models, the silhouettes of foreground objects are extracted from the video sequences. Each pixel of the segmented regions is expressed with two bits: the first bit indicates the group of the pixel, and the second bit represents the intra-reliability of the result as shown in Table 1.

**TABLE 1. Categories of silhouette masks**

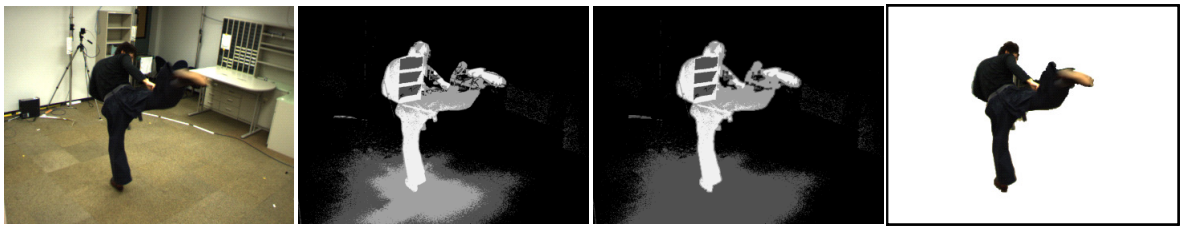
| Code | Group | Intra-reliability |
|---|---|---|
| 01 | Background | Reliable |
| 00 | Background | Suspicious |
| 10 | Foreground | Suspicious |
| 11 | Foreground | Reliable |

First, initial region classification is performed by subtracting the luminance components of the current frame from the background model. The initial object region is classified into four categories using multiple thresholds based on background subtraction *BD*, as in Eq. (3). $L_I$ and $L_B$ indicate the luminance components of pixel *p* in the current frame and the background model, respectively, and *b* is a scale parameter of the background model:

$$BD(p) = \left| L_I(p) - L_B(p) \right| \quad (3)$$

$$\begin{cases} BD(p) < K_1 b(p) & \Rightarrow \text{Reliable Background} \\ K_1 b(p) \le BD(p) \le K_2 b(p) & \Rightarrow \text{Suspicious Background} \\ K_2 b(p) \le BD(p) \le K_3 b(p) & \Rightarrow \text{Suspicious Foreground} \\ K_3 b(p) \le BD(p) & \Rightarrow \text{Reliable Foreground} \end{cases}$$

Then, the suspicious regions from the initial classification are refined using a hue component because the shadow or lighting change the color properties of the background much less than the luminance. The system applies Eq. (3) to the hue component in a similar manner with single parameter $K_4$ and refines the suspicious regions. The absolute of difference in Eq. (3) should be an angle distance due to the angular nature of the hue component. The eliminated regions are changed to



(a) Original image (b) Initial regions (c) Shadow elimination (d) Final result

**Fig. 2. Results of silhouette extraction.** In the results, the black, dark gray, light gray and white regions indicate Reliable background, Suspicious background, Suspicious foreground and Reliable foreground, respectively.

*Suspicious background* regions.

Thresholds $K_1$~$K_4$ are determined by training data with the following condition, where $\beta$ was empirically set to 3 because false positive errors are generally more critical than false negative errors in 3D reconstruction:

$$(K_1, K_2, K_3, K_4) = \underset{K_1,K_2,K_3,K_4}{\arg\min} \begin{pmatrix} \beta \times False\ Negative\ Error \\ + False\ Positive\ Error \end{pmatrix} \quad (4)$$

Finally, the final foreground is extracted using a silhouette extraction technique that wraps the object with four drapes to smooth the foreground boundaries and eliminate holes inside the regions [36]. Newly covered background regions by silhouette extraction are changed to *Suspicious foreground* regions. Figure 2 shows the results of silhouette extraction in each step.

**Results.** Figure 3 shows the segmentation results of various scenes: the left image shows the captured image, and the right image shows the extracted foreground in each pair. Figure 4 shows objective evaluations of the foreground extraction algorithm. We randomly selected 14 frames from six different scenes (i.e., 84 images in total) and created ground-truth segmentation masks by manual segmentation. Then we compared the segmentation error of the proposed algorithm with a Gaussian-based algorithm with a single threshold [37] and a KDE-based algorithm [18]. We applied the same morphological processes to all experiments to see the effect of the proposed model. We compared the results by calculating the percentage of erroneous pixels as Eq. (5).

$$error = \frac{Number\ of\ erronous\ pixels}{Number\ of\ real\ foreground\ pixels} \times 100\ (\%) \quad (5)$$

The silhouette extraction algorithm works in 6~7 frames/secs with XGA image sequences. The detailed analysis for processing time is dealt in Section 4 because the balance with other processes in the whole system should also be evaluated.

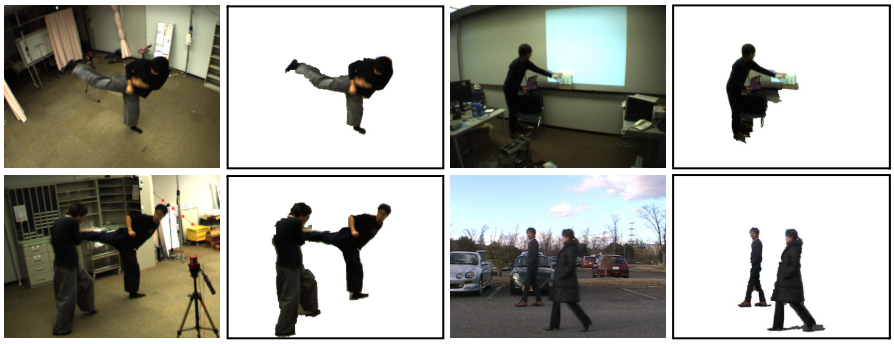


**Fig. 3. Foreground extraction in various environments.**

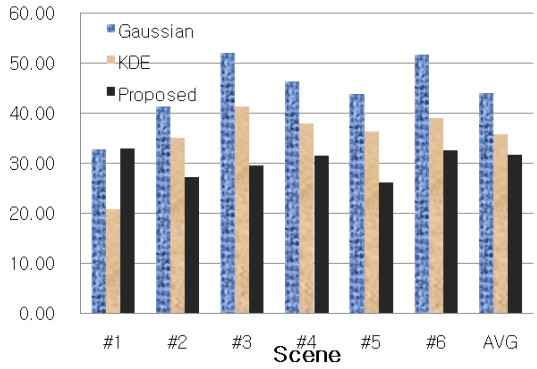



**Fig. 4. Segmentation errors to ground-truth (%)**

### 3.2. Voxel Reconstruction

The SFS technique carves 3D space by projecting silhouettes from each view into the space using projection matrixes obtained by camera calibration. The visual hull constructed using SFS provides an upper bound on the object shape. This system uses an improved SFS technique to reconstruct a visual hull in the presence of outliers. Here, we considered two kinds of outliers that can happen in real environments: segmentation errors and partial occlusions [6][38]. The proposed algorithm compensates for modeling errors due to the outliers by considering both the intra- and inter-reliabilities of the silhouettes that represent the reliability of the segmentation itself and the reliability from the projection circumstances to the other cameras, respectively.

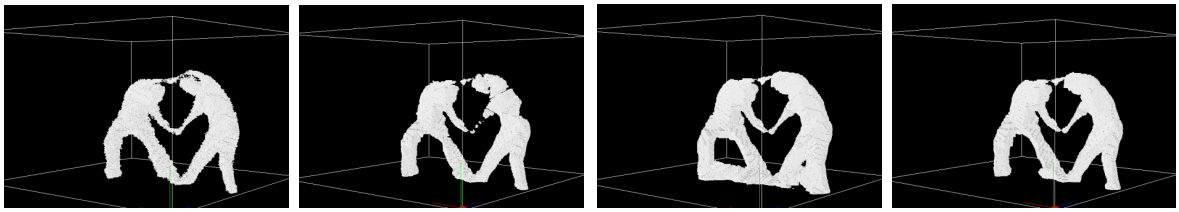
In the SFS process, a single segmentation error classified into the background with low intra-reliability can damage the visual hull and nullify all correct information in the other images. On the other hand, an occluder in front of the target object at the view of a certain camera has a high possibility to be a Reliable background region because it is unaffected by the foreground object. However, we must exercise caution when determining whether it is a partial occluder or a real hole in the space. Therefore, we establish a set of rules to distinguish outliers in silhouette masks, where $m_p$ is a projected point of 3D point $M(X,Y,Z)$ on a silhouette mask $S_p$ $(p=1,\ldots,N)$.

- if ($m_p$==00 && first bit (all other $m_n(n \neq p)$)==1)
  then $m_p$=10 *//Segmentation error*
- if ( $m_p$==01 && $m_{p-1}$==11 && $m_{p+1}$==11)
  then $m_p$=10 *//Occluded region*

**Implementation on GPU.** Visual hull reconstruction using the SFS method is extremely time-consuming because this method checks all voxels in the modeling space by projecting them into all silhouette image planes. Due to the highly parallel nature of the voxel reconstruction process, the SFS algorithm would function better on a platform that supports the parallel processing of data. To this end, we implemented the algorithm on a GPU. Visual hull reconstruction is implemented using a single fragment shader, and results are rendered directly to a three-dimensional texture using a framebuffer object (FBO) one layer at a time. Each layer is rendered such that the fragment shader is run once per texel.

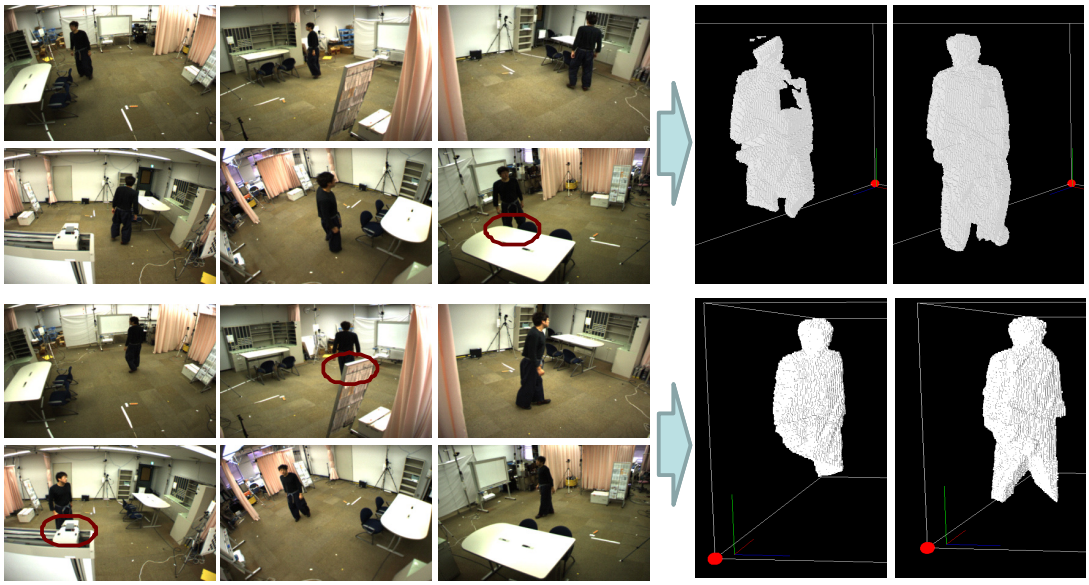
Our implementation uses two textures: one to store the silhouette images as input, and another to store the reconstructed 3D voxel model. For each frame, the silhouette images are loaded into a three-dimensional texture in the video memory as a stack of two-dimensional images. Then we use a fragment shader to execute the reconstruction algorithm on each voxel in the three-dimensional output texture. However, modern hardware is currently only capable of rendering two-dimensional images. As such, we divide the voxel space into two-dimensional slices of data and render them one at a time. The output texture is specifically sized to ensure that the fragment shader is run once for each voxel. That is, each pixel that comes from the fragment shader represents a single voxel.

**Results.** Figure 5 shows snapshots of generated visual hulls. The visual hull was damaged by segmentation errors in Fig. 5(b) that were caused by a conventional method. Therefore, we loosened the segmentation parameters to compensate for cracks, but this coarsened the visual hull, as seen in Fig. 5(c). On the other hand, the visual hull generated based on reliability looks much closer to the real 3D model. Figure 6 shows the results in an environment where occluders exist. The result was affected by segmentation errors and occlusion by a table, a rack, or a whiteboard. However, the proposed method constructed more natural models by compensating for the outliers. In the second test set, the legs are not restored perfectly because their lower part is occluded by two different occluders. We restricted the maximum number of partial occluders to one per voxel in this experiment. It can be changed to deal with multiple occlusions, but the model recovered from more than two occluders may become too coarse due to a lack of information; moreover it roughens the other part of the visual hull and increases computational complexity.



(a) (b) (c) (d)

**Fig. 5. Reconstructed visual hulls.** (a) With ground-truth masks; (b) Conventional SFS with optimized parameters; (c) Conventional SFS with loosened parameters; (d) Proposed method with reliabilities



(a) (b) (c)

**Fig. 6. Results with partial occlusion.** (a) Captured videos; (b) Conventional method; (c) proposed method with reliabilities

### 3.3. Virtual View Rendering

**Microfacet billboarding.** In our system, a microfacet billboarding technique is employed to solve the limitations of mesh-based approaches [39]. Microfacet is defined as "a slice which intersects the center of the voxel and is vertical to the viewing direction." Thus, as the virtual camera is rotated around the model, the normals of each rendered microfacet are adjusted to be parallel to the virtual camera's view vector. The second reason that we employ microfacet billboarding is that it is very suitable for GPU characteristics. The system does not need to calculate the normal vectors of facets against a virtual camera because the geometry shader works with a camera coordinate.

**Camera selection and texturing.** Due to occlusion and changes in a surface's appearance caused by lighting conditions and viewing angles, great care must be taken to select the appropriate camera to use as a texture for each microfacet. Improper camera selection can lead to unrealistic images due to

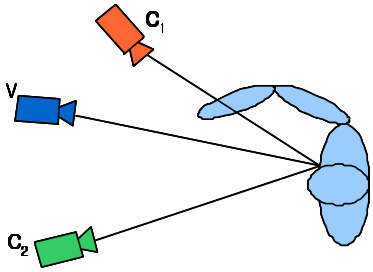



**Fig. 7. Occlusion problem in camera selection.**

**Fig. 8. Example of depth textures. Intensity of each pixel represents its depth; lighter parts are further away than darker parts.**

distortion, clipping, or occlusion.

To select the most suitable texture for rendering, the system chooses three candidate cameras per microfacet that minimize the following energy of texture:

$$E_{TEX} = \alpha \times \left( \frac{| \theta_p |_{(\mathrm{mod}360)}}{360} \right) + \beta \times d_p + \gamma \times reliability(m_p) \qquad (6)$$

The first factor is the primary one that investigates the angles between the virtual camera and environmental camera $C_p$. If the angle is larger than 90°, the first factor is set $\infty$ because it cannot be projected on the right side of the facet. The second factor is related to the resolution of textures to provide the most detailed texture where $d_p$ denotes a distance from environmental cameras $C_p$ to a facet. Finally the system refers to the reliability of silhouette masks because the wrong texture might be mapped from segmentation errors or occluders. Weighting factors $\alpha$, $\beta$, and $r$ are determined empirically.

**Occlusion handling.** The above method's primary problem, however, is that it ignores the possibility of occlusion. When non-trivial objects are recorded by the proposed system, the model generated often occludes itself in one or several of the captured camera images. Figure 7 demonstrates this problem. Notice that the texture of the hand will be rendered on the chest if $c_1$ is selected for texturing

in the virtual view $V$. Only proper occlusion detection can eliminate this problem.

Given viewpoint $V$, voxel $M_n$ is considered occluded if voxel $M_o$ lies directly between $M_n$ and $V$. Thus, one method of checking for occlusion is to use a three-dimensional version of Bresenham's line drawing algorithm to obtain the coordinates of every voxel between $V$ and $M_n$ [40]. However this method requires many accesses into the voxel model. Alternatively, avoiding so many lookups into the voxel model is possible with some precomputation. Before the model is rendered from the virtual viewpoint, depth texture $T_p$ is rendered from the perspective of each camera. The value of texel $t$ in $T_p$ represents the distance from $C_p$ to the closest microfacet along the ray projected from $C_p$ through $t$. Examples of the depth textures generated by this method are shown in Fig. 8.

Thus, when rendering, it is possible to use these depth textures to check for occlusion. If the distance between voxel $V$ and camera $C_p$ is larger than the value in the depth map, it means that the voxel is occluded from the camera. Then the algorithm continues to check the remaining cameras based on their priority until one is found that is not occluded.

Precomputing depth textures requires an additional step before rendering, but it reduces the computational complexity of the occlusion verification for each voxel to a single texture check and comparison.

**Implementation on GPU.** To implement microfacet billboarding, our system primarily relies on the geometry shader because it operates in a camera space. This means that the microfacet vertices are trivial to compute, since the process involves the addition of fixed offsets in the $x$ and $y$ directions. This ensures that the resulting microfacet will always directly face the camera.

In camera selection and texturing, the vertex and geometry shaders cooperate with each other. Depth textures are generated by rendering all space-occupying voxels as microfacets from the perspective of a given camera. We set each microfacet's color equivalent to its distance from the camera. Thus, the final texture will contain the distance to the closest microfacet at each texel. The vertex shader begins by checking the voxel texture to ascertain whether the voxel needs to be rendered. If it does, the voxel's position in the global coordinate system is calculated. Then, the vertex shader sorts the list of cameras and picks the best three by calculating the energy of texture in Eq. (6). The geometry shader receives this list along with the voxel information and calculates and calculates the distance between the voxel and the cameras. The shader continues execution with the other selected cameras until it finds one that is not occluded. Finally, the selected camera's image is used to texture the microfacet.

**Results.** Figure 9 shows generated micro-facet billboards and selected cameras for each facet. The texture in the body is partly replaced with pink cameras because the microfacets are occluded from the blue camera by the left hand.

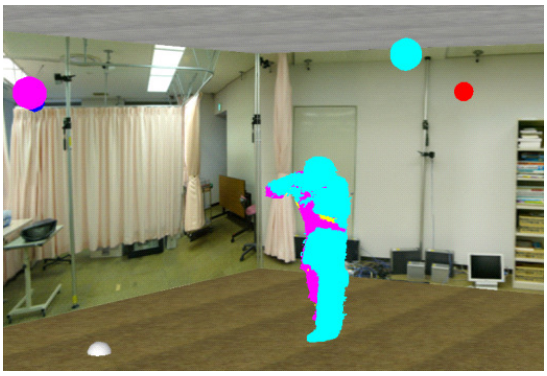


**Fig. 9. Results of microfacet generation and camera selection. Surface color corresponds to the color of each camera.**

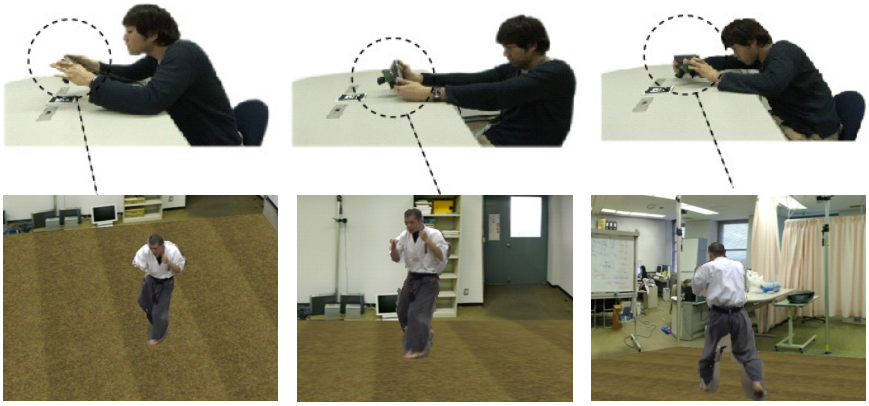
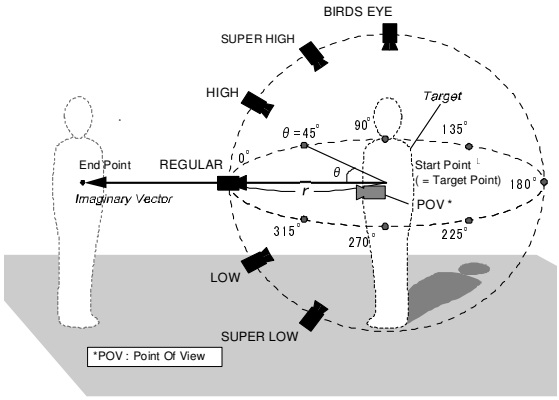


**Fig. 10. Real-time rendering with a USB camera and ARToolkit. We attached a small display behind the USB camera to match camera direction and user's viewpoint.**

Users can watch the rendered results in real-time by controlling a virtual camera with a keyboard or ARToolkit. ARToolkit tracks the relative position of the fiducial marker to the camera and synthesizes the reconstructed 3D model on the marker. We also attached a small display behind a normal USB camera to match the camera direction and the user's viewpoint. Therefore, the users control the marker and the USB camera as if on a miniature set of 3D space with a real camera to shoot the scene. Figure 10 shows scenes where a user is controlling the USB camera and the generated video from the designated camera position.

## 4. CINEMATOGRAPHIC VIRTUAL CAMERA CONTROL

In this section, we introduce our cinematographic camera controlling system to produce movie-like footage from the reconstructed 3D models. The grammar of film language is based on a constrained condition for switching sequential camera shots. Since each single shot is nothing more than a video fragment, many shots must be combined to generate an entire video in a sequential combination of shots called a "scene."




(a) Initial camera parameters

| Class | Shot name |
|---|---|
| Fixed | FixShot, BustShot, MediumShot, LongShot |
| Moving independently | CraneUpShot, CraneDownShot, RaiseUpShot, SpinAroundShot, TimeSliceShot |
| Moving tied to target | PanShot, DollyShot |
| Zoom | ZoomOutShot, ZoomInShot,WhipZoomShot |

(b) Taxonomy of camera actions

**Fig. 11. Camera shots**

## 4.1. Camera Shots

Camera shot information is generally comprised of two types of information for camera control: initial camera parameters and camera actions (Fig. 11). The initial camera parameters are set to appropriately capture the target objects in the initial state. They specifically describe the relative angle between a target object and a capturing camera, in addition to the object's size and position in the captured image. The camera action describes variations in the camera position and the zoom parameters. In the proposed system, values can be set in the following two ways: using the time-series relative difference from the initial state and calculating the values by interpolating the initial and exit states, given as input information. The proposed system provides fourteen sets of camera actions.

## 4.2. Annotation

To apply the shots to the scene, information about the positions of the target objects and the capturing-time code is necessary. We use annotation information that consists of spatial and temporal information.

Figure 12 shows a screenshot of our developed interface tool for simultaneously inputting spatial and temporal annotations. Here, a spatial annotation is defined by dragging a mouse over the area, while a temporal annotation is defined by clicking on a certain point on a time-scale bar. These annotations are recorded with index information and an extra "user's area" described in a free format. Annotations are assumed to be made not only by humans but also by various sensor inputs such as IR and pressure sensors.

## 4.3. Applying Camera Shots

The system generates a scene by declaring a set of camera shots that follow the rules in the grammar of film language. If a suitable set of camera shots is found among the preserved scene

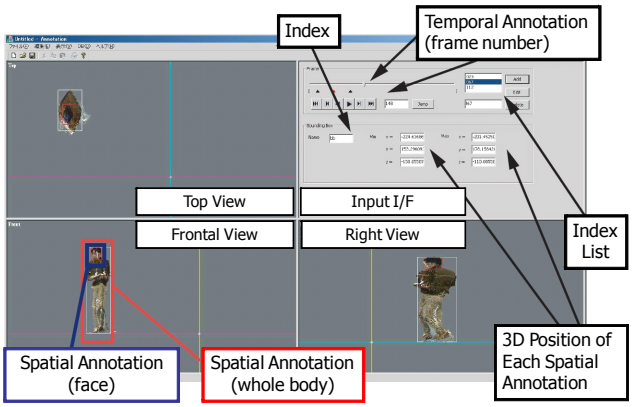



**Fig. 12. Interface of developed annotation tool**

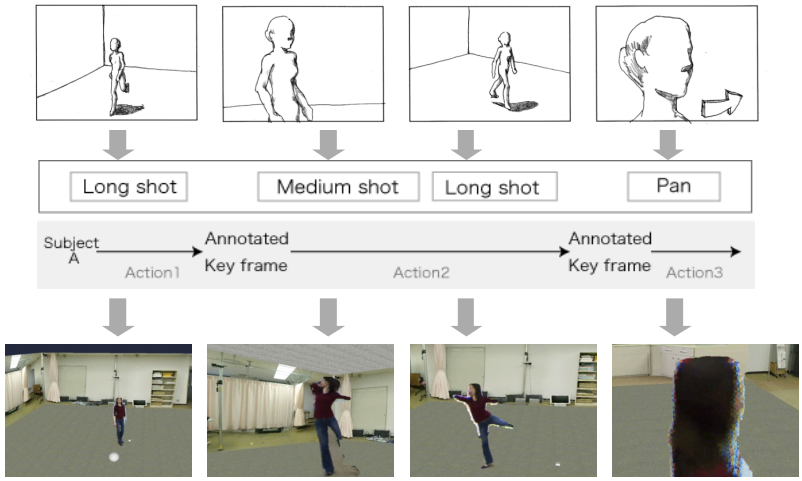



**Fig. 13. Example of scene production**

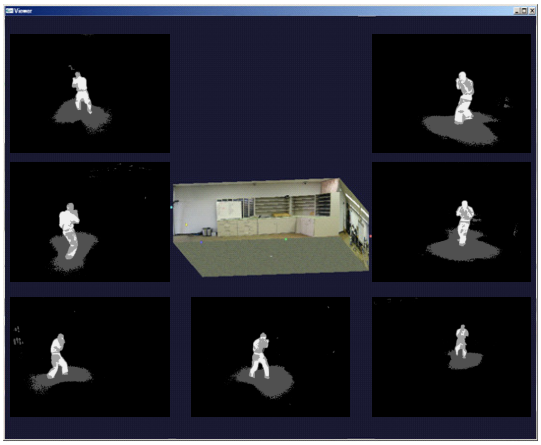
databases of experts, the user can apply the set to his/her model by setting the camera's initial positions and key frames. If the user cannot find a suitable set of shots, however, the system creates a new set of camera shots and stores it in the database.

Figure 13 shows an example of applying camera shots to the 3D models with the camera control system. If the user defines the target objects, key frames, camera positions, and camera actions, then the system generates a final video based on a storyboard.

When users make a new set of shots, the system helps produce the most natural and cleanest scene by providing information to them. First, it warns the user if he/she tries to combine shots that violate the grammar of film language. For example, the constrained condition for switching camera shots explains the contraindication of continuity between a current shot and the next shot. The system provides the following two contraindications of camera shot switching based on the grammar of film language.

1. Do not set the next camera shot to cross the imaginary line because it confuses the audience.
2. Do not choose a subsequent camera shot that resembles the current shot because the similarity reduces the effectiveness of the switch.

(a) Captured multiple videos



(b) Silhouette and reliability information

(c) Scenes from arbitrary viewpoints

**Fig. 14. Results of free-view generation.**

The system also shows the sum of the texture energy of Eq. (6) in the scene when the user selects an initial position of the virtual camera and key frames on the timeline. Rendered scenes show very different quality of textures based on the position of the virtual camera because the environmental cameras are sparsely set up. For example, a scene from a point between two cameras in the distance shows more distorted texture than one from a closer position to an environmental camera. By checking the energy functions in the scene based on the virtual camera's position, the user can decide the best shots from the candidates being considered.

# 5. FINAL RESULTS

## 5.1. Free-view Generation

We tested the proposed system in various indoor places, in various times of a day, and with various people. Visual hulls were constructed at a resolution of 300x300x200 on a 1cmx1cmx1cm regular

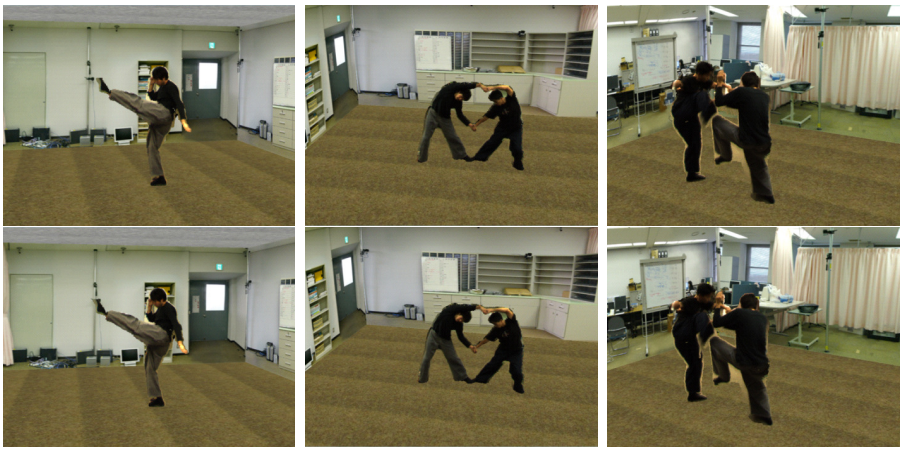


**Fig. 15. Snapshots of generated free-view stereo videos.** (top) Left images; (bottom)Right images

voxel grid using the proposed reconstruction algorithm.

For objective evaluation, we made ground-truth 3D model with the manually segmented masks, and calculated errors of reconstructed models by conventional visual hull method with Gaussian-based segmentation [1] and results with the proposed algorithm. Table 2 shows voxel errors of the reconstructed 3D models against the ground-truth model. We tested randomly-selected 14 frames in the sequence and calculated errors in the same manner as Eq. (5) in three dimensions. The false negative (FN) error whereby the 3D objects is falsely classified as null space normally comes from occlusion, and the false positive (FP) errors from shadows or redundancy of the reliability-based visual hull. The proposed algorithm shows much lower error rate than the conventional visual hull algorithm, because the algorithm creates more tight segmentation masks and compensates 3D models occluded from certain camera view.

**Table 2. Voxel errors to ground-truth model (%)**

| | FP | FN |
|---|---|---|
| Conventional | 95.24 | 9.56 |
| Proposed | 29.72 | 4.45 |

Figure 14 shows the captured multiple videos, silhouette masks with reliabilities, and final rendered images from arbitrary viewpoints. The system can also produce stereoscopic 3D videos by dual rendering. Figure 15 shows snapshots of the free-viewpoint stereo videos rendered from the visual hulls using two virtual cameras aligned in parallel. Natural 3D scenes from any point of view can be rendered from the proposed system though it shows degraded texture in zoomed-in scenes because of the limited resolution of the fixed environmental cameras.

**Run-time analysis.** We selected 200 frames of multiple video sets from various scenes that contained just one foreground object, computed the average processing time for a single frame of data to

reconstruct and render one full 3D voxel model in Table 3.

The whole system ideally works at about 5~6 frames/sec. However, in a practical rendering system connected to camera clients by a network, the speed is almost halved because the process also needs idle time to gather the complete information necessary to make 3D models from all camera clients. The frame rate of the whole system also depends on the complexity of the objects

**Table 3. Speed analysis of system (msec)**

| Capturing | | Modeling and Rendering | |
|---|---|---|---|
| Stage | Time | Stage | Time |
| Capturing | 31.22 | Data transfer | 6.73 |
| Subtraction | 23.57 | 3D reconstruction | 179.77 |
| Labeling | 6.70 | Depth texturing | 0.89 |
| Silhouette extraction | 73.15 | Rendering | 0.77 |
| Background update | 15.02 | | |
| Total | 149.66 | Total | 188.16 |
| Speed (fps) | 6.68 | Speed (fps) | 5.31 |

## 5.2. Cinematographic Video Production

Finally, the system generates a video by piecing together all the 3D videos generated from designated shots. Figure 16(a) shows example footage of a 3D video with camera works using two spatial and five temporal annotations. Varied shots with different angles and framings are set to dramatically show a karate player. Figure 16(b) shows another piece of footage from the same 3D model to which different shots are applied. In the upper-left of Fig. 16(a), the virtual camera position is near the ceiling even though in Fig. 16(b) the virtual camera targets his foot. These pieces of footage were made from the same 3D model, but they give rather different impressions. In Fig. 16(a), the CraneDownShot shot gives the initial state as “Long” framing and a “BIRDS EYE” angle, and then the camera position is gradually moved closer to the ground. In Fig. 16(b), on the other hand, the DollyShot shot maintains a fixed distance to the annotated target, the foot, using “Close” framing and the “REGULAR” angle. Figure 17 shows storyboards that indicate the locations of the virtual camera, the shot names, and the parameters in the footage of Fig. 16. These pieces of footage indicate that the user can aim the camera to generate a dramatic video, as shown in Fig. 16(a), or a suspenseful one, as in Fig. 16(b). Our system can obviously produce video to respond to user requests.

After saving these sets of camera shots with new names “Dramatic” and “Suspense,” we captured new contents in different environment. After reconstructing the visual hulls of the new scene, we loaded the sets from the database of the camera control system and applied them to the new model. We simply annotate the foreground objects and set the key frames. Then the system automatically produces new videos with the sets of camera shots. Figure 18 shows the results of the “Ballet” scene with the same camera shots.

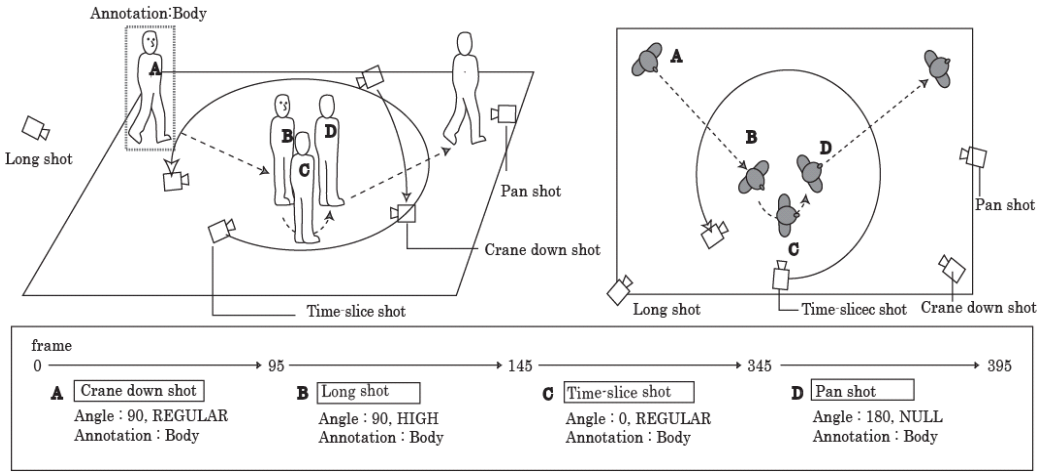
### 5.3. Applying to Daily Lives

We tested the proposed system in various real environments. The system is relatively small, so it is easy to set up at any place. We mounted seven synchronized IEEE cameras to surround the capture space and connected them to capturing PCs. In order to get the camera parameters, we calibrated all cameras with a checker board and a 3D laser-surveying instrument [41], but it can be replaced with simpler solutions such as wand calibration [31]. After checking the system in real-time mode, we recorded the scene as uncompressed videos for off-line processing. In the off-line processing, we reconstructed 3D model for each frame and saved them again as voxel sequence to apply camera works. Finally we rendered a video from the 3D model using the cinematographic virtual camera control system.

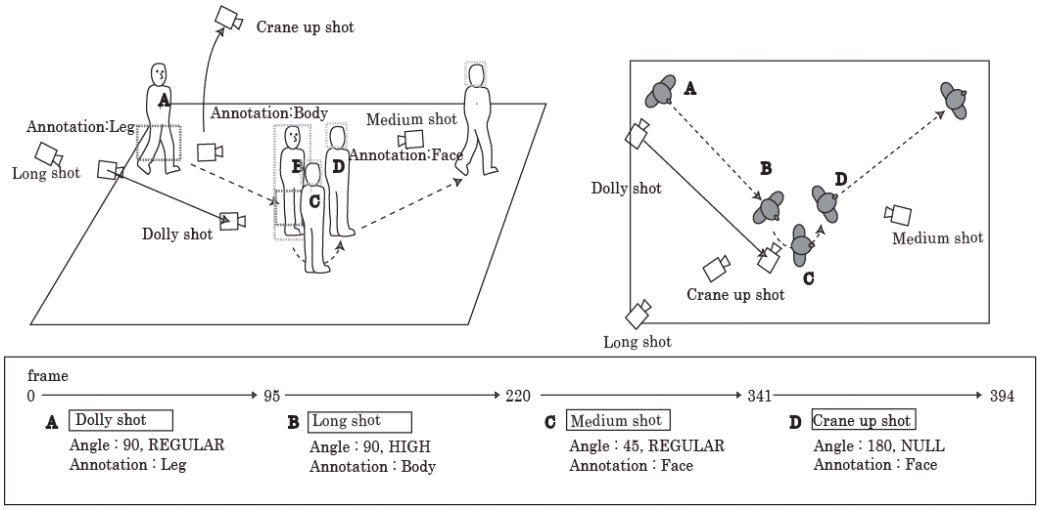
Figure 19 shows snapshots of the rendered videos. First, we captured scenes in the same place for long time and generated models of people's activity in the space. It was open space so that the background and brightness vary according to time. We updated the background model for silhouette extraction using the similar way to Elgammal's method [18]. Second, we also tested the system with various environment and conditions. We assumed very challenging situations such as occlusions between multiple people, interaction with small props and occlusions from large objects such as table and chairs. Therefore, we can see some errors of models and textures in the final scene, and some users may not be satisfied with the quality of the produced final video comparing with other time-consuming state of the art methods [31][42]. However, all scenes were produced with only seven small cameras and four normal PCs, and it took just several minutes from capturing to final rendering including application of camera works for a video of 15 seconds. If we need to increase the quality of model and textures for the final video, we can apply more time-consuming refinement processes such as global or local optimization of 3D models [5, 31, 42] as post-processing.

(a) "Dramatic" shots (b) "Suspense" shots

**Fig. 16. Produced scenes of a karate player**




(a) Dramatic




(b) Suspense

**Fig. 17. Overview of camera works in Fig. 16**

(a) “Dramatic” shots (b) “Suspense” shots

**Fig. 18 Reproduced “Ballet” scenes with same camera works**

(a) (b)

**Fig. 19. Produced cinematographic free-view videos in our daily lives.** (a) for many hours at the same place; (b) from various environments and situations.

Video clips displaying all processes and results from the proposed system can be downloaded from the following address.

http://www.3dkim.com/research/CR.mov or http://www.3dkim.com/research/CR.wmv

## 6. CONCLUSION AND FUTURE WORK

The ultimate goal of this study is to record our real lives and display them as movie-like footage. We presented a practical method to construct the visual hull from multiple videos in a real environment and to display realistic images of those objects from arbitrary viewpoints. The system was realized to work in near real-time with a few ordinary PCs. Then we applied expert knowledge to generate desirable and interesting film footage using a sequence of shots taken by virtual cameras. Due to the simple interface of the proposed camera control system, even non-experts can easily convert their own 3D model to attractive movies that look as if they were edited by experienced filmmakers with the help of the system's expertise. The applications can be extended to surveillance, remote training or education, and entertainment fields as well as personal life recording.

Although significant advances have been made in this paper, there are still many areas that need to be explored in future research. In terms of camera calibration, it is necessary to devise more handy and fast calibration tool. Bundle adjustment [43] or wand calibration [31] which considers the inter-camera relationship between cameras can be a solution to increase efficiency and the accuracy of the calibration. In silhouette extraction, we have to develop a more powerful segmentation algorithm which is robust to variations in the background and extract fine silhouettes. The current algorithm

works well for indoor scenes, but it is still hard to be applied to outdoor scenes because of the strong shadows by the sun and unstable background. We can also see flickering on the boundaries of moving object because temporal consistency is not considered. We are going to devise algorithm to track the stable object boundaries on 2D silhouette and 3D model over frames. Finally, future works should include improving the system speed to full real-time and developing a texture-blending algorithm from multiple images to generate more natural surface texture.

Although the system still has problems to be overcome, we believe that the proposed system offers a framework for popularizing 3D imaging systems and applications.

# References


[1] T. Kanade, P.W. Rander and P.J. Narayanan, "Virtualized Reality: Constructing Virtual Worlds From Real Scenes," *IEEE Multimedia*, vol. 4, no. 1, pp.34-47, 1997.

[2] M.A. Magnor, *Video-Based Rendering*, A K Peters, 2005.

[3] 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. SIGGRAPH*, pp. 819-827, 2003.

[4] K. Grauman, G. Shakhnarovich and T. Darrell, "A Bayesian Approach To Image-Based Visual Hull Reconstruction," *Proc. CVPR,* pp. 187-194, 2003.

[5] J.-Y. Guillemaut, A. Hilton, J. Starck, J. Kilner and O. Grau, "A Bayesian Framework for Simultaneous Matting and 3D Reconstruction," *Proc. 3DIM*, pp. 167 – 176, 2007.

[6] L. Guan, J.S. Franco and M. Pollefeys, "3D Occlusion Inference From Silhouette Cues," *Proc. CVPR*, pp. 1-8, 2007.

[7] 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*, vol. 14, no. 3, pp. 357-369, 2004.

[8] M. Ueda, D. Arita and R. Taniguchi, "Real-Time Free-Viewpoint Video Generation Using Multiple Cameras and a PC-Cluster," *Proc. PCM, LNCS 3331*, pp. 418–425, 2004.

[9] M. Li, M. Magnor and H.P. Seidel, "Improved Hardware-Accelerated Visual Hull Rendering," *Proc. Vision, Modeling, and Visualization,* 2003.

[10] W.E. Lorensen and H.E. Cline, "Marching Cubes: A High Resolution 3D Surface Constructing Algorithm," *Proc. SIGGRAPH*, pp.163-169, 1987.

[11] A. Gueziec and R. Hummel, "Exploiting Triangulated Surface Extraction Using Tetrahedral Decomposition," *IEEE TVCG*, vol. 1, no. 4, pp. 328-342, 1995.

[12] E.D. Gelasca, T. Ebrahimi, M. Karaman and T. Sikora, "A Framework for Evaluating Video Object Segmentation Algorithms," *Proc. CVPR Workshop*, pp.198, 2006.

[13] M. Piccardi, “Background Subtraction Techniques: A Review,” *Proc. IEEE SMC*, vol. 4, pp. 3099-3104, 2004.

[14] C. Stauffer and W.E.L. Grimson, “Adaptive Background Mixture Models for Real-Time Tracking,” *Proc. CVPR*, pp.246-252, 1999.

[15] D.S. Lee, J.J. Hull, and B. Erol, “A Bayesian Framework For Gaussian Mixture Background Modeling,” *Proc. ICIP*, vol. 3, pp. 973-976, 2003.

[16] F. Porikli and O. Tuzel, “Human Body Tracking by Adaptive Background Models and Mean-Shift Analysis,” *Proc. Pets-ICVS,* 2003.

[17] O. Tuzel, F. Porikli and P. Meer, “A Bayesian Approach To Background Modeling,” *Proc. IEEE MVIV*, vol.3, pp. 58-63, 2005.

[18] A. Elgammal, D. Harwood and L.S. Davis, “Non-Parametric Model For Background Subtraction,” *Proc. ECCV*, vol. 2, pp. 751-767, 2000.

[19] B. Han, D. Comaniciu and L. Davis, “Sequential Kernel Density Approximation Through Mode Propagation: Applications To Background Modeling,” *Proc. ACCV,* 2004.

[20] A. Mittal and N. Paragios, “Motion-Based Background Subtraction Using Adaptive Kernel Density Estimation,” *Proc. CVPR,* pp. 302-309, 2004.

[21] J. Ahn and H. Byun, “Accurate Foreground Extraction Using Graph Cut With Trimap Estimation,” *Proc. PSIVT*, pp. 1185-1194, 2006.

[22] M. Sormann, C. Zach and K. Karner, “Graph Cut Based Multiple View Segmentation for 3D Reconstruction,” *Proc. 3DPVT*, pp. 1085 – 1092, 2006.

[23] B. Furht (Ed.), *Encyclopedia of Multimedia*, 2nd ed., pp.170-171, Springer, 2008.

[24] D. Arijon, *Grammar of the Film Language*, Silman-James Press, 1991.

[25] S.D. Katz. *Cinematic Motion: Film Directing: A Workshop for Staging Scenes*, Michael Wiese Film Productions, 2004.

[26] L. He, M.F. Cohen and D.H. Salesin, “The Virtual Cinematographer: A Paradigm For Automatic Real-Time Camera Control And Directing,” *Proc. ICCGIT*, pp. 217-224, 1996

[27] A. Inoue, H. Shigeno, K. Okada and Y. Matsushita, “Introducing Grammar of the Film

Language into Automatic Shooting for Face-To-Face Meetings," *Proc. SAINT,* pp. 277-280, 2004.

[28] R. Ronfard and G. Taubin, "Introducing 3d Cinematography," *IEEE CG&A*, vol. 27, no. 3, pp. 18-20, 2007.

[29] R. Tenmoku, R. Ichikari, F. Shibata, A. Kimura and H. Tamura, "Design And Prototype Implementation Of Mr Pre-Visualization Workflow," *Proc. ISMAR Workshop for Mixed Reality Entertainment and Art*, 2007.

[30] T. Takai, S. Nobuhara, H. Yoshimoto and T. Matsuyama, "3D Video Technologies: Capturing High Fidelity Full 3D Shape, Motion, and Texture," *Proc. ISMAR Workshop,* 2006.

[31] J. Starck and A. Hilton, "Surface Capture for Performance-Based Animation," *IEEE CG&A,* vol. 27, no. 3, pp.21-31, 2007.

[32] F. Fabiz, A. Cheok, L. Wei, Z. Zhiying, X. Ke, S. Prince, M. Billinghurst and H. Kato, "Live Three-Dimensional Content for Augmented Reality," *IEEE Tans. Multimedia*, vol. 7, no. 3, pp. 514-523, 2005.

[33] M.S. Allili, N. Bouguila and D. Ziou, "Finite Generalized Gaussian Mixture Modeling and Applications to Image and Video Foreground Segmentation," *Proc. CRV*, pp. 183 – 190, 2007.

[34] H. Kim, R. Sakamoto, I. Kitahara, T. Toriyama and K. Kogure, "Robust Foreground Extraction Technique Using Gaussian Family Model and Multiple Thresholds," *Proc. ACCV,* 2007.

[35] J.Y. Lee and A.K. Nandi, "Maximum Likelihood Parameter Estimation of the Asymmetric Generalized Gaussian Family of Distribution," *Proc. SPW-HOS,* 1999.

[36] P. Kumar, K. Sengupta and S. Ranganath, "Real Time Detection and Recognition of Human Profiles Using Inexpensive Desktop Cameras," *Proc. ICPR,* pp.1096-1099, 2000.

[37] C. Wren, A. Azarbayejani, T. Darrell and A.P. Pentland, "Pfinder: Real-Time Tracking of the Human Body," *IEEE Trans. PAMI*, vol. 19, no. 7, pp.780-785, 1997.

[38] H. Kim, R. Sakamoto, I., Kitahara, T. Toriyama and K. Kogure, "Reliability-Based 3D Reconstruction in Real Environment," *Proc. ACM Multimedia*, pp. 257-260, 2007.

[39] S. Yamazaki, R. Sagawa, H. Kawasaki, K. Ikeuchi and M. Sakauchi, "Microfacet Billboarding," *Proc. Eurographics Workshop on Rendering*, pp. 175–186, 2002.

[40] J. Bresenham, “Algorithm for Computer Control of a Digital Plotter,” *IBM Systems Journal,* vol. 4, no. 1, pp. 25-30, 1965.

[41] H. Kim, I. Kitahara, R. Sakamoto and K. Kogure, “An Immersive Free-Viewpoint Video System Using Multiple Outer/Inner Cameras,” *Proc. 3DPVT*, June 2006.

[42] S. Nobuhara and T. Matsuyama “Deformable Mesh Model for Complex Multi-Object 3D Motion Estimation from Multi-Viewpoint Video,” *Proc. 3DPVT*, pp.264-271, 2006.

[43] B. Triggs, P. McLauchlan, R. Hartley and A. Fitzgibbon, “Bundle Adjustment - A Modern Synthesis,” *Proc. International Workshop on Vision Algorithms: Theory and Practice*, pp.298-372, 1999.

# Affiliation of author

Hansung Kim
Centre for Vision, Speech and Signal Processing in Univercity of Surrey, UK.
h.kim@surrey.ac.uk

**Hansung Kim** received the BS degree in radio communication engineering in 1998, and the MS and Ph.D degrees in electronic and electrical engineering from Yonsei University, Korea, in 2001 and 2005, respectively. He was employed as a researcher of Knowledge Science Lab (KSL) at Advanced Telecommunications Research Institute International (ATR), Japan, from 2005 to 2008. He is currently a research fellow of Centre for Vision, Speech and Signal Processing (CVSSP) in University of Surrey, UK. His research interests include three dimensional image processing, computer vision, and mixed reality. He is a member of IEEE.

Ryuuki Sakamoto
Wakayama University, Japan
rkskmt@sys.wakayama-u.ac.jp

**Ryuuki Sakamoto** received the BS degree in Information Science in 1997 from Okayama University of Science, Japan, and MS and PhD degrees in Knowledge Science at Japan Advanced Institute of Science and Technology (JAIST). Since 2003, He has been a researcher in the Knowledge Science Lab (KSL) at Advanced Telecommunications Research Institute International (ATR), Japan. He is currently an assistant professor at Wakayama University. His research interests include CSCW and Context-aware computing. He is a member of ACM, Information Processing Society of Japan (IPSJ), and Japan Creativity Society (JCS).

Itaru Kitahara
University of Tsukuba, Japan
kitahara@iit.tsukuba.ac.jp

**Itaru Kitahara** received the B.E. and M.E. degrees in Science Engineering from University of Tsukuba, Japan in 1994 and 1996, respectively. In 1996 he joined Sharp Corporation. From 2000 to 2003, he was a research associate of the Center for Tsukuba Advanced Research Alliance, University of Tsukuba. He received the PhD degree in Systems and Information Engineering from University of Tsukuba in 2003. Since 2003, he has been a researcher at ATR. From 2005, he has been an assistant professor at University of Tsukuba. He was awarded the IEEE VR (Conference on Virtual Reality) Honorable Mention Award in 2003. His research interests include computer vision, mixed reality and intelligent image media. He is a member of IEEE.

Tomoji Toriyama
Toyama Prefectural University
toriyama@pu-toyama.ac.jp

**Tomoji Toriyama** received B.E. and M.E. degrees from Toyama University and a Ph.D. from Toyama Prefectural University, Japan. In 1987, he joined Nippon Telegraph and Telephone Corporation (NTT) at Kanagawa. From 2005-2008, he was a researcher at ATR, Japan. Since 2008, he has been a professor at the Toyama Prefectural University. His research interests include man-machine interface, and image processing.

Kiyoshi Kogure
Knowledge Science Lab, ATR, Kyoto, 619-0288, Japan
kogure@atr.jp

**Kiyoshi Kogure** received his PhD in engineering from Keio University. He is the director of the ATR Knowledge Science Laboratories. His research interests include intelligent environments, intelligent agents, and natural language processing.