World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:2, No:8, 2008

Real-time 3D Feature Extraction without Explicit 3D Object Reconstruction

Kwangjin Hong, Chulhan Lee, Keechul Jung, and Kyoungsu Oh

***Abstract*—**For the communication between human and computer in an interactive computing environment, the gesture recognition is studied vigorously. Therefore, a lot of studies have proposed efficient methods about the recognition algorithm using 2D camera captured images. However, there is a limitation to these methods, such as the extracted features cannot fully represent the object in real world. Although many studies used 3D features instead of 2D features for more accurate gesture recognition, the problem, such as the processing time to generate 3D objects, is still unsolved in related researches. Therefore we propose a method to extract the 3D features combined with the 3D object reconstruction. This method uses the modified GPU-based visual hull generation algorithm which disables unnecessary processes, such as the texture calculation to generate three kinds of 3D projection maps as the 3D feature: a nearest boundary, a farthest boundary, and a thickness of the object projected on the base-plane. In the section of experimental results, we present results of proposed method on eight human postures: T shape, both hands up, right hand up, left hand up, hands front, stand, sit and bend, and compare the computational time of the proposed method with that of the previous methods.

***Keywords*—**Fast 3D Feature Extraction, Gesture Recognition, Computer Vision.

I. Introduction

THE recognition algorithm is significant to the interactive computing environment. Additionally, the processing time and recognition accuracy are the main concerns of the recognition algorithm. Therefore, various researches related to these concerns have been studied in the last few years.

Generally, computer vision-based recognition algorithms use 2D images for extracting features. The 2D images can be used efficiently when the camera position and viewing direction are fixed. The features, extracted from 2D input images, are invariant to the scale, the translation, and the rotation in 2D planes. However, in spite that the targets, which are captured and recognized, are 3D objects, the features, which are extracted in 2D images, can have 2D information or limited 3D information.

This work was supported by the 'Seoul R and BD Program (10581cooperateOrg93112)'.

K. Hong is with the Department of Media, Graduate school of Soongsil University, Seoul, Korea (e-mail: hongmsz@ ssu.ac.kr).

C. Lee is with the Department of Media, Graduate school of Soongsil University, Seoul, Korea (e-mail: dashans @ssu.ac.kr).

K. Jung is with the Department of Media, Graduate school of Soongsil University, Seoul, Korea (corresponding author to provide phone: 82-2-812-7520; fax: 82-2-822-3622; e-mail: kcjung@ ssu.ac.kr).

K. Oh is with the Department of Media, Graduate school of Soongsil University, Seoul, Korea (e-mail: oks@ ssu.ac.kr).

To solve this problem, a lot of studies proposed methods using multi-view images[1,2,3]. These methods recognize objects or postures using comparison results between camera input images and multi-view camera captured images which are captured by real or virtual cameras around the objects of recognition. However, these methods spend very long time to generate features and to compare the features with input data, since the accuracy of recognition is proportional to the number of camera view images. And the major problem of these methods is that the features extracted from multi-view images cannot fully represent the 3D information. This is due to those images still containing only the 2D information.

Therefore, recently a lot of studies are proposing many kinds of methods using reconstructed 3D objects. The reconstructed 3D objects can represent positions of components which include 3D objects and can provide 3D information to extracted features. Therefore, these ways can recognize more accurate than the methods which use the 2D images. Table I shows the kinds of computer vision-based feature extraction methods using the reconstructed 3D objects.

TABLE I
The Vision-Based 3D Feature Extraction Methods Using Reconstructed 3D Objects

Type of Extracted Features	Algorithms for Feature Extraction	Authors[paper]	Feature Extraction Time(sec)
Histogram	3D Bin-distribution	C. Chu, I. Cohen [4]	Less than 0.1
		D. Kyoung et al. [5]	Less than 1
	Spherical harmonic	T. Funkhouser et al. [6]	Less than 1
Graph	Reeb graph	M. Hilaga et al. [7]	1
	3D thinning	H. Sundar et al. [8]	10
	Curve-skeleton	N. D. Cornea et al. [9, 10]	10^3
		A. Brennecke, T. Isenberg [11]	10^3

The methods using the structural feature of 3D objects are more accurate for recognition, because these extract the features using 3D information of each component that constructs the subjects of recognition (Table I). However, the methods producing skeletons from 3D objects [7-11] required long time, since these are divided into two processes: the 3D object reconstruction and the feature extraction. To solve this problem in the feature extraction part, the methods, which use a spherical harmonic [6] or a 3D bin-distribution algorithm [4,5] for fast feature extraction and represent distances between the center point and the boundary point by a histogram, is proposed. However, even though these methods can represent the global

shape, they cannot represent the local characters. Due to the 3D object reconstruction part still exists, it is difficult to apply these methods using 3D objects to the real-time recognition environment.

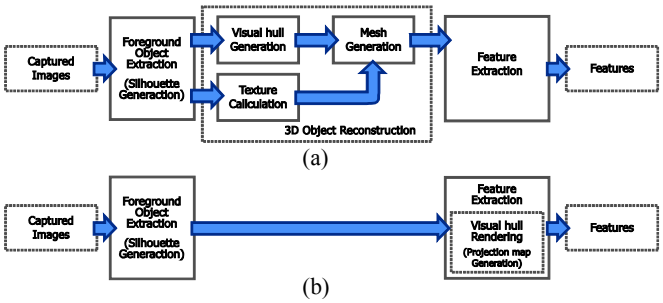


Fig. 1 The overviews of 3D feature extraction processes: (a) process of previous studies and (b) proposed method

In this paper, we propose the method of a real-time 3D feature extraction without the explicit 3D object reconstruction. Fig. 1 shows the difference between the previous feature extraction methods and the proposed one in their processes. This method can generate three kinds of features which contain different types of 3D information: nearest boundary, farthest boundary, and thickness of the object projected on a base-plane. The projection map can be obtained by rendering the target object. For this purpose, the visual hulls can be used as a 3D geometry proxy. It is an approximate geometry representation resulting from the shape-from-silhouette 3D reconstruction method[12].

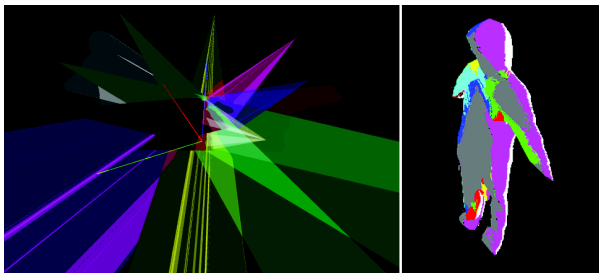
The visual hull reconstruction and rendering can be accelerated by modern graphics hardware. Li et al.[13] present a hardware-accelerated visual hull (HAVH) rendering technique. Since we extract features from the results of the visual hull rendering, our proposed method does not need explicit geometric representation. Therefore we use the modified HAVH algorithm which disables unnecessary processes, such as the texture calculation, in the general HAVH algorithm. Moreover, we can save the drawing time by disabling all lighting and texture calculations for this rendering, due to these processes are not necessary for feature extraction (Fig. 1(b)).

The structure of the paper is as follows. We describe the visual hull in Section II. Next, we describe the details of our methods in Section III: the silhouette extraction (Section III.A), the visual hull rendering (Section III.B) and the projection map generation (Section III.C). Experimental results are provided in Section IV. And, we conclude in Section V.

II. Visual Hull

For extracting features of dynamic 3D objects, we can use video streams or images as input from multiple cameras, and reconstruct an approximate shape of the target object from multiple images. By rendering the reconstructed object, we are able to obtain projection maps, which can be used as important features of the object. For the purpose of reconstructing and visualizing the dynamic object, the visual hull can be used. It has been widely used as 3D geometry proxy, which represents a conservative approximation of true geometry [12].

We can reconstruct a visual hull of an object with calibrated cameras and the object's silhouette in multiple images. The silhouette of the object in an input image refers to the contour separating the target object from the background. Using this information, combined with camera calibration data, the silhouette is projected back into the 3D scene space from the cameras' center of projection. This generates a cone-like volume (silhouette cone) containing the actual 3D object. With multiple views, these cones can be intersected. This produces the visual hull of the object (Fig. 2).



(a) (b)

Fig. 2 Visual hull reconstruction: (a) 8 silhouette cones are generated from silhouette images taken from different viewpoints, (b) Reconstructed 3D surface

Many different implementations of visual hull reconstruction are described in the literature [13-16]. Some compute an explicit geometric representation of the visual hull, either as voxel volume [14] or polygonal mesh [15]. However, if the goal is rendering visual hulls from novel viewpoints, the reconstruction does not need to be explicit. Li et al. [13] present a hardware-accelerated visual hull (HAVH) rendering technique. It is a method for rendering of visual hull without reconstructing the actual object. The implicit 3D reconstruction is done in rendering process by exploiting projective texture mapping and alpha map trimming. It runs on modern graphics hardware and achieves high frame rates.

We can obtain projection maps for feature extraction by rendering the visual hull. The explicit geometry representation is not needed for this process. Moreover, explicit geometry reconstruction is very time-consuming. Instead of reconstructing 3D visual hull geometry, we render the visual hull directly from silhouettes of input images by using HAVH method and obtain the projection maps from the rendering results.

III. Fast Feature Extraction

For extracting features of a dynamic 3D object, we render a visual hull of the target object from multiple input images. By using HAVH rendering method, we can render the visual hull without reconstructing the actual object in real time. From the rendering results of the visual hull, we obtain projection maps which contain 3D information of the target object, such as nearest boundary, farthest boundary, and thickness of the object (Fig. 3). They can be used as important features of the target object.

The projection maps are obtained by rendering the target object. When an object is rendered by a 3D graphics card, the depth of a generated pixel is stored in a depth buffer. The depth buffer can be extracted and saved as a texture [17], called a depth map. By rendering the front-most surfaces of the visual hull, we can get a depth map which stores the distance from a projection plane to the nearest boundary. It is called a nearest boundary projection map (Fig. 3(a)). Likewise, we can get a farthest boundary projection map by rendering the rear-most surfaces of the visual hull (Fig. 3(b)). By subtracting the values from the two maps, we can get a thickness map which stores the distance between the front-most surfaces and rear-most surfaces (Fig. 3(c)).

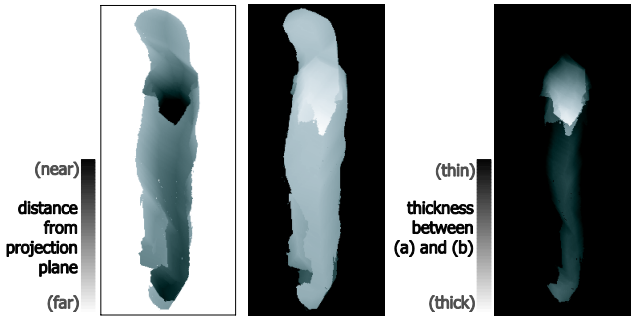


Fig. 3 Projection maps: (a) nearest boundary projection map, (b) farthest boundary projection map, and (c) thickness map

Our method consists of two major parts as shown in Fig. 4. When images are captured from cameras, an object's silhouette can be extracted in the multiple images. Using this information, combined with calibration data, we can render the visual hull of the target object. We are able to obtain projection maps of the object while rendering the visual hull.

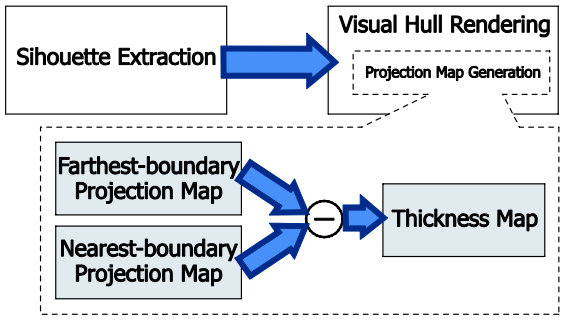


Fig. 4 Work flow of our method.

A. Silhouette Extraction

When images are captured from multiple cameras, an object's silhouette can be computed in multiple images. The target object in each captured image(I_c) is segmented from the background (I_b). We store the information into silhouette images(S). The alpha values of a silhouette image are set to 1 for the foreground object and to 0 for the background as in (1).

$$S(x,y) = \begin{cases} 1 & \text{if } |I_c(x,y) - I_b(x,y)| > thresho \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Silhouettes are then generated from each silhouette image. The silhouette of the object in a silhouette image refers to the collection of all edges separating the foreground object from the background. Using this information, combined with calibrated cameras, we are able to generate silhouette cones by projecting back each silhouette into 3D scene space.

B. Visual Hull Rendering

The visual hull surfaces can be determined on graphics hardware by exploiting projective texturing in conjunction with alpha blending while rendering silhouette cones. As shown by Fig. 5(a), for rendering a silhouette cone of the n^{th} camera, all silhouette images(S_1, S_2, …, S_{n-1}) except the one associated with the cone currently being drawn are projected onto the silhouette cone of C_n using the projection matrices from their corresponding calibrated cameras. These silhouette images are used as a mask eliminating the portions of each cone that do not lie on the surface of the visual hull. In the texture units, alpha values projected from multiple textures are modulated. As a result, only those pixels projected with the alpha value 1 from all the other silhouette images produce the output alpha value 1(Fig. 5(b)). Thus, visual hull faces are drawn. All polygons of silhouette cones are still rendered entirely, but using the alpha testing, only the correct parts of them actually generate pixels in the image.

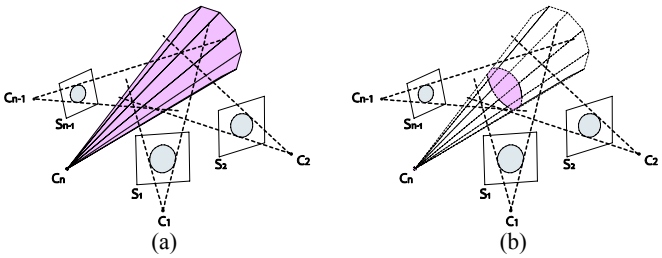


Fig. 5 Silhouette cone rendering: (a) while rendering each silhouette cone, it is protectively textured by the silhouette images from all other views. (b) alpha map trimming, alpha values from multiple textures are modulated. Thus, visual hull faces are drawn

C. Projection Map Generation

We can compute the distance from a projection plane to front-most surfaces of a target object as well as the distance to rear-most surfaces. We are then able to compute the thickness of the object, which is the distance between front-most surfaces and rear-most surfaces. Consider the example in Fig. 6. Given a vector perpendicular to the projection plane, we can find hitpoints: P_1 on the front-most surface and P_2 on the rear-most surface. The distance between P_1 and P_2 can be computed. It equals to $||P_2-P_1||$.

We can generate the projection map by rendering the target object. First, we set a virtual camera to be able to view the 3D object. The object from a viewpoint is then projected onto the camera's view plane (or projection plane). An orthographic

projection can be used in order to avoid perspective projection distortion. Rasterization, which is the process of converting geometric primitives into pixels, determines the viewing direction and its hitpoint. In rendering the object's front-most surface, the hitpoint P_1 on the front-most surface along the viewing direction is easily extracted for each pixel and saved in a buffer. Likewise, the hitpoint P_2 on the rear-most surface can be obtained by re-rendering the object from the same viewpoint and saved in another buffer. With the information from the two buffers, we can compute the distance.

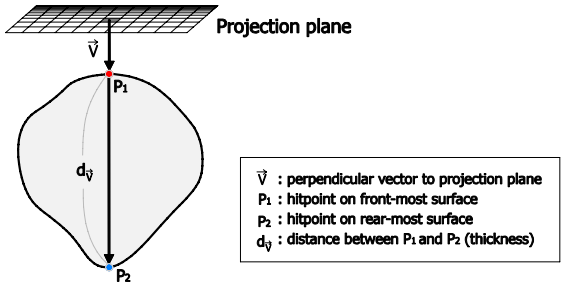


Fig. 6 Distance from a projective plane to front-most of an object, distance to rearmost surfaces, and its thickness

For the implementation, we can generate projection maps using depth information from the viewpoint by rendering the target object. When an object is rendered by a 3D graphics card, the depth of a generated pixel is stored in a depth buffer. It is done in hardware. The depth buffer can be extracted and saved as a texture, called a depth map. It is usual to avoid updating the color buffers and disable all lighting and texture calculations for this rendering in order to save drawing time. We render the target object from a viewpoint with the depth test reversed (i.e., GL_GREATER instead of GL_LESS) in order to draw the rear-most faces of the object. From this rendering, the depth buffer is extracted and store in a texture, which is a farthest boundary map (Fig. 7(a)). To obtain a nearest boundary map, we render the object again from the same viewpoint with the normal depth test only passing fragments closer to the viewpoint (i.e. GL_LESS) (Fig. 7(b)). We can compute the distance by subtracting the values from the two depth buffers in order to generate a thickness map. It can be done by multiple textures blending function (i.e., GL_SUBTRACT)(Fig. 7(c)).

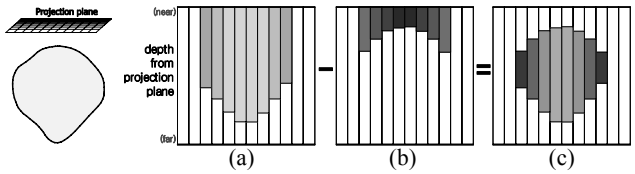


Fig. 7 Projection map generation using depth map. (a),(b), and (c) are 1D version of projection maps of an object shown in left: (a) farthest boundary projection map stores the depth from view plane to rear-most surface, (b) nearest boundary projection map stores the depth values of front-most surface, (c) thickness map is generated by subtracting (b) from (a)

IV. Experimental Results

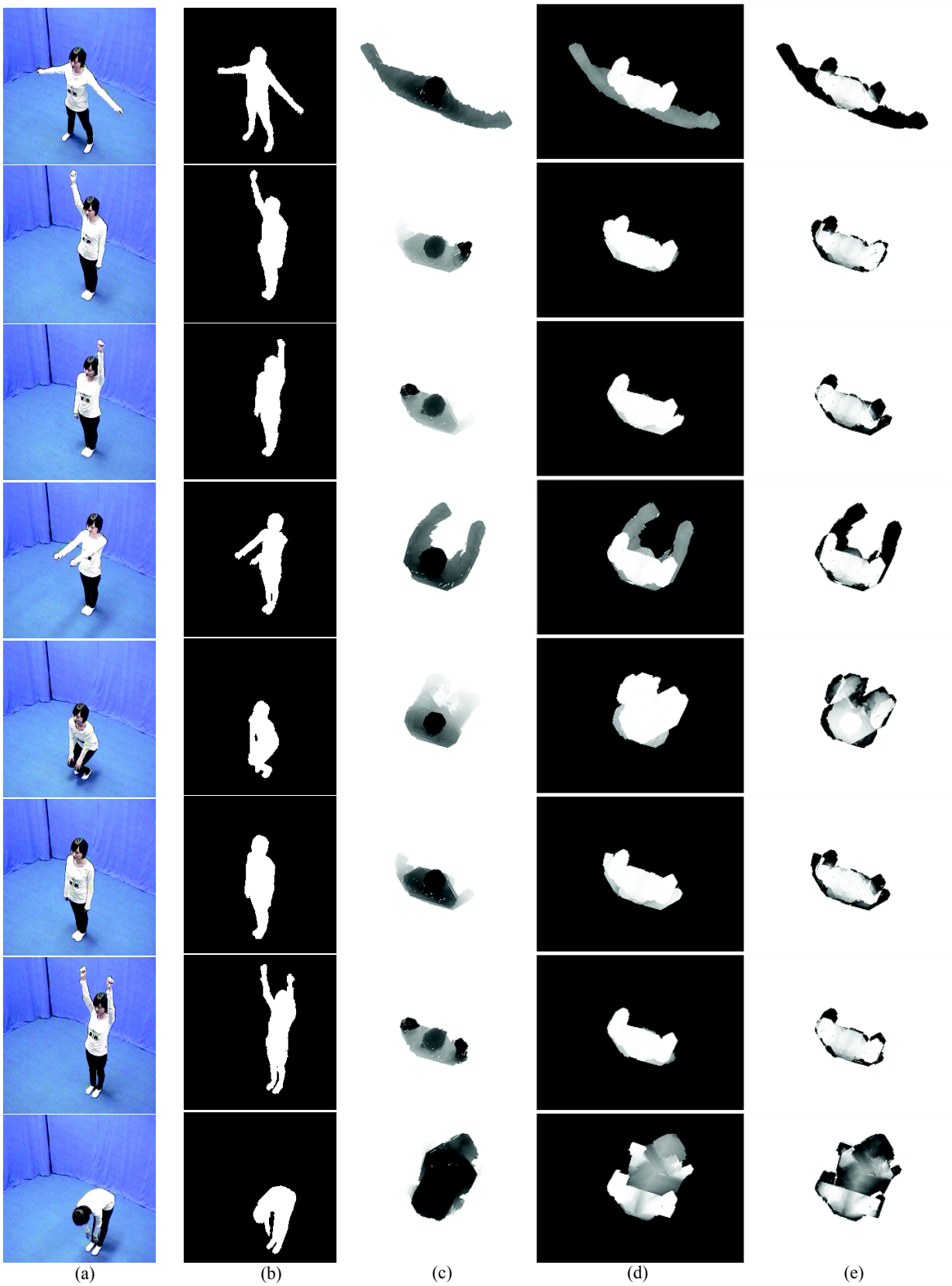
This section demonstrates our results of the fast feature extraction. All images have been generated on a 2.13GHz CPU with 2Gbyte memory and an nVidia GeForce 8800GTX graphic card, using Direct3D with HLSL. We used 8 cameras to acquire input images. The cameras were positioned around an object in an accurately calibrated system. The resolution both acquired images and rendered result images was set to 640 480. Under this setting, we have measured the speed of our method. We obtained 8 silhouette cones from silhouette images. It took around 8ms per image on the CPU. However, we did not check the calculation time of this process (generating silhouette cones), due to this is a common factor for all algorithm. Generating a single projection map by rendering front-most (or rear-most) surfaces of the visual hulls, which is the process of a nearest (or furthest) boundary projection map, took around 1.5ms. The generation times for a thickness map including the generation of two projection maps and distance computation by rendering the visual hull twice were about 3ms (Table II).

TABLE II
The Comparison of the Proposed Method with the 3D Feature Extraction Methods Which Use Explicit 3D Models

Using Methods	Visual Hull Generation	Feature Extraction	Total
Thinning-based Skeletonization	370ms	10^7 ms	10^7 ms
3D bin-distribution	370 ms	10 ms	380 ms
Proposed method	3 ms		3 ms

Experimental results show that the proposed method provides high accuracy of recognition and fast feature extraction. Table II shows the comparison of the proposed method with the 3D feature extraction methods which use explicit 3D models. For this experiment, we use the 3D models that are reconstructed in the voxel space of 300x300x300 size, and that are generated by GPU. Because we generate the projection map using GPU programming without explicit 3D object reconstruction, the proposed method is faster than other methods and can manage 13 or 14 image sets per second. Therefore this method is affected to real-time recognition system.

Fig. 8 shows the silhouette images which are extracted only foreground objects in a camera captured image (Fig. 8(a)) and projection maps which are generated using the reconstructed 3D objects. In this paper, we use 8 kinds of human posture images. And, the projection maps are generated using a top-view camera with an orthographic projection. Because the human postures are limited to the z=0 plane and the top-view image is invariant to the translation, scale and rotation, we use the top-view image. As shown by Fig. 8(b), there are many similar silhouette images in different posture and different camera views. However, the projection maps can represent the difference of each posture, since they have the 3D information of each posture (Fig. 8(c-e)).



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

Fig. 8 Extracted features from the captured images of 8 human postures: (a) camera captured images, (b) silhouette images, (c) nearest boundary projection map, (d) farthest boundary projection map and (e) thickness map

V. Conclusion

In this paper, we proposed a 3D feature extraction method without the explicit 3D object reconstruction. The proposed method generates 3 kinds of projection maps, which project all data on the z=0 plane using the input images of the multi-view camera system, instead of 3D object. This method is fast for presenting the 3D information of the object in input images, due to we use the modified HAVH algorithm that the unnecessary processes are disabled such as the light and texture calculation. Therefore the proposed method can apply to real-time recognition system. However, some problems remain in this method: error in visual hull rendering, limitation of the number of camera, data transferring time in memories and distance calculating between overlapping components.

In our method, we use the silhouette-based visual hull rendering algorithm. But this algorithm cannot generate the accurate 3D object, because the silhouette images are binary images and does not have the input object's texture information. And our method cannot use more than 16 camera images. However, this is a hardware limitation and we can solve this problem using parallel visual hull rendering method. Finally, the proposed method cannot detect the z-position of arms or legs, because we calculate only the distance between the nearest and the farthest parts from a camera. Now we are studying about reducing transfer time and more accuracy to provide good performance.

References

[1] J. Loffler, "Content-based retrieval of 3d models in distributed web databases by visual shape information," in *Proc. 4th International Conf. Information Visualization*, 2000, pp. 82.

[2] C.M. Cyr, B.B. Kimia, "A similarity-based aspect-graph approach to 3d object recognition," *International J. Computer Vision,* vol. 57, 2004, pp. 5-22

[3] P. Min, J. Chen, T. Funkhouser, "A 2d sketch interface for a 3d model search engine," in *Proc. the International Conf. Computer Graphics and Interactive Techniques,* 2002, pp. 138.

[4] C. Chu, I. Cohen, "Posture and gesture recognition using 3d body shapes decomposition," in *Proc. the IEEE Computer Society Conf. CVPR,* vol. 3, 2005, pp. 69.

[5] D. Kyoung, Y. Lee, W. Baek, E. Han, J. Yang, K. Jung, "Efficient 3d voxel reconstruction using pre-computing method for gesture recognition," in *Proc. Korea-Japan Joint Workshop,* 2006, pp. 67-73.

[6] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, D. Jacobs, "A search engine for 3d models," *ACM Trans. Graphics*, vol. 22, 2003, pp. 83-105.

[7] M. Hilaga, Y. Shinagawa, T. Kohmura, T. Kunii, "Topology matching for fully automatic similarity estimation of 3d shapes." in *Proc. the 28th annual Conf. Computer graphics and interactive techniques,* 2001, pp. 203-212.

[8] H. Sundar, D. Silver, N. Gagvani, S. Dickinson, "Skeleton based shape matching and retrieval," in *Proc. International Conf. Shape Modeling International,* 2003, pp. 130-139.

[9] N.D. Cornea, D. Silver, P. Min, "Curve-skeleton properties, applications and algorithms," *IEEE Trans. Visualization and Computer Graphics*, vol. 13, 2007, pp. 530-548.

[10] N.D. Cornea, D. Silver, X. Yuan, R. Balasubramanian, "Computing hierarchical curve-skeletons of 3d objects," in *Proc. the Visual Computer*, vol. 21, 2005, pp. 945-955.

[11] A. Brennecke, T. Isenberg, "3d shape matching using skeleton graphs," in *Proc. Simulation and Visualization,* vol. 13, 2004, pp. 299-310

[12] A. Laurentini, "The visual hull concept for silhouette-based image understanding," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, 1994, pp. 150-162.

[13] M. Li, M. Magnor, H. Seidel, "Hardware-accelerated visual hull reconstruction and rendering," in *Proc. Graphics Interface*, 2003, pp. 65-71.

[14] R. Szeliski, "Rapid octree construction from image sequences," in *Proc. CVGIP: Image Underst.*, vol. 58, 1993, pp. 23-32.

[15] W. Matusik, C. Buehler, L. McMillan, "Polyhedral visual hulls for real-time rendering," in *Proc. the 12th Eurographics Workshop. Rendering Technique*, 2001, pp. 115-126.

[16] C. Lee, J. Cho, K. Oh, "Hardware-accelerated jaggy-free visual hulls with silhouette maps," in *Proc. the ACM Sym. Virtual Reality Software and Technology*, 2006, pp. 87-90.

[17] C. Everitt, A. Rege, C. Cebenoyan, "Hardware shadow mapping," *Technical report*, NVIDIA.