3D Object Extraction from 2D Object

M. Ramasubramanian\(^1\)  M.A. Dorai Rangaswamy\(^2\)

\(^1\)Research Scholar & Assistant Professor Grade -II, Aarupadai Veedu Institute of Technology, Rajiv Gandhi Salai,(OMR), Department of Computer Science and Engineering, Paiyanoor-603104, Kancheepuram District, Tamilnadu, India

mailtoraams@gmail.com Ph:09884415533

\(^2\)Research Supervisor & Sr. Professor, Vinayaka Missions University, Aarupadai Veedu Institute of Technology, Rajiv Gandhi Salai,(OMR), Department of Computer Science and Engineering, Paiyanoor-603104, Kancheepuram District, Tamilnadu, India

drdorairs@yahoo.co.in Ph:09940345654

Abstract - The gesture recognition is to be studied vigorously, for the communication between human and computer in an interactive computing environment. Lots of studies have proposed the efficient methods about the recognition algorithm using 2D camera captured images. There is a limitation to these methods are exist, such as the extracted features cannot represent fully the object in the real world.

Keywords: 3D object Extraction, Gesture Recognition, Computer Vision.

Though, 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 the related researches. Therefore we are proposing a new method to extract the 3D features combined with the 3D object reconstruction. Our propose method uses the enhanced GPU-based visual hull generation algorithm.

This algorithm disables unnecessary processes, like the texture calculation, a nearest boundary, a farthest boundary, and a thickness of the object projected on the base-plane. In the experimental section results, we are presenting the results of proposed method on ten human postures. T shape, both hands up, right hand up, left hand up, hands front, bend with hands up, bend with hands front, stand, sit and bend, and compare the computational time of the proposed method with that of the previous methods.

I. INTRODUCTION

The recognition algorithm is much significant to the interactive computing environment. In addition to that, the processing time and recognition accuracy are the main factors of the recognition algorithm. Hence, various researches related to these concerns have been studied in the last decade. Generally, computer vision-based recognition algorithms will use 2D images for extracting features. The 2D images can be used efficiently, when the position of the camera and viewing direction are fixed. The extracted features from 2D input images, are invariant to the scale, the translation, and the rotation in 2D planes. However, in spite that the targets, that are captured and recognized, are 3D objects, the features, which are extracted in 2D images, that can have 2D information or limited 3D information. In order to solve this problem, a lot of studies proposed using multi-view images. These methods will recognize the objects or postures using comparison results between camera input images and multi-view camera captured images that are captured by real or virtual cameras around the objects of recognition.

However, these methods spends very long time to generate features and to compare the features with the 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 the images still containing only the 2D information. So, 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

<table>
<thead>
<tr>
<th>Type of Extracted Features</th>
<th>Algorithms for Feature Extraction</th>
<th>Authors [paper no.]</th>
<th>Feature Extraction Time in seconds</th>
</tr>
</thead>
<tbody>
<tr>
<td>Graph</td>
<td>Reeb graph</td>
<td>M. Hilaga et al. [8]</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>3D thinning</td>
<td>H. Sundar et al. [9]</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>Curve-skeleton</td>
<td>N. D. Cornea et al. [10,11]</td>
<td>103</td>
</tr>
</tbody>
</table>

ISBN NO: 978-93-80609-14-0
The methods using the structural featuring of 3D objects are more accurate for recognition, because these extract the features using 3D data of each component that constructs the subjects of recognition (Table I). However, the methods producing the skeletons from 3D objects [8-12] requires the long time, since these are divided into two processes: the 3D object reconstruction and feature extraction.

To solve this problem, in the feature extraction part, the methods, which use a spherical harmonic [7] or a 3D bin-distribution algorithm [5,6] for fast feature extraction and represent the 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. 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.

![Fig. 1 The 3D feature extraction processes:](image)

(a) process of existing studies and (b) proposed method

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[13].

The visual hull reconstruction algorithm and rendering can be accelerated by the modern graphics hardware. Li et al.[14] presented a hardware-accelerated visual hull (HAVH) rendering technique. As we are extracting features from the results of the visual hull rendering, our proposed method does not need explicit geometric representation. Therefore we use the enhanced HAVH algorithm which disables unnecessary processes, such as the calculation of texture, in the general HAVH algorithm. Moreover, we can save the drawing time by disabling all lighting and texture calculations for this rendering. 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). The Experimental results are provided in Section IV. And, we conclude in Section V.

II. VISUAL HULL

For extracting the features of dynamic 3D objects, we can use the 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 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 the calibrated cameras and the object’s silhouette, in multiple images. The shadow or 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 the 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

---

<table>
<thead>
<tr>
<th>Histogram</th>
<th>Isenberg [12]</th>
<th>Less than 0.1</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D Bin-distribution</td>
<td>C. Chu, I. Cohen [5]</td>
<td>Less than 1</td>
</tr>
<tr>
<td>Spherical harmonic</td>
<td>T. Funkhouser et al. [7]</td>
<td>Less than 1</td>
</tr>
</tbody>
</table>
Visual hull reconstruction: Reconstructed 3D surface

Different implementations of visual hull reconstruction are described in the literature [14-17]. Some of them compute an explicit geometric representation of the visual hull, either as voxel volume [15] or polygonal mesh [16]. However, if the goal is rendering visual hulls from novel viewpoints, the reconstruction does not need to be explicit. Li et al. [14] present a hardware-accelerated visual hull (HAVH) rendering technique. It is a method for rendering of visual hull without reconstruction of the actual object. The implicit 3D reconstruction is done in rendering process by exploiting projective texture mapping and alpha map trimming. It runs on the 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 process. 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

In order to extract the 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 the real time. From the rendering results of the visual hull, we can obtain the 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 the 3D graphics card, depth of a generated pixel is stored in a depth buffer. The depth buffer can be extracted and will be saved as a texture [18], called a depth map. By rendering the front-most surfaces of the visual hull, we will get the 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 will 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.

![Projection maps: (a) nearest boundary projection map, (b) farthest boundary projection map, and (c) thickness map](image)

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.

![Work flow of our method](image)

A. Silhouette Extraction: When the images are captured from multiple cameras, an object's silhouette can be computed in multiple images. The target object, in each captured image(Ic) is segmented from the background (Ib). We can 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).

\[ S(x, y) = \begin{cases} 
1 & \text{if } I_c(x, y) - I_b(x, y) > \text{threshold} \\
0 & \text{otherwise}
\end{cases} \] (1)
Silhouettes or shadows 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 the 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 nth camera, all

The silhouette images of S1,S2,S3, …,Sn-1 are used as the mask, eliminating the portions of each cone, that do not lie on the surface of the visual hull. In the texture unit, the 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 will produce the output alpha value 1(Fig. 5(b)). Thus, the visual hull faces are drawn. All the polygons of silhouette cones are still rendered entirely, but using the alpha testing, only the correct parts of them are actually generating the pixels in image.

![Fig. 5 Silhouette cone rendering](image)

To implement, we can generate projection maps using depth information from the viewpoint by rendering the target object. When an object is rendered by 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 in order to draw the rear-most faces of the object. From this rendering, the depth buffer is extracted and stored in a texture, which is a farthest boundary map (Fig. 7(a)). To obtain a nearest boundary map, we have to render the object, again from the same viewpoint with the normal depth test by passing the fragments closer to the viewpoint (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 (Fig. 7(c)).

![Fig. 6 Distance from a projective plane to front-most of an object, distance to rearmost surfaces, and its thickness](image)
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 2.13 GHz CPU with 2Gbyte memory and an nVidia GeForce 8800GTX graphic card, using Direct3D. We used ten cameras to acquire input images. The cameras were positioned in and around the object accurately calibrated system. The resolution of both acquired images and rendered result images was set to 640X480. Under this setting, we have to measure the speed of our method. We obtained the ten 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, due to this is a common factor for all algorithms. 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 farthest boundary projection map, took around 1.50 ms. 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 3.0 ms (Table II).

<table>
<thead>
<tr>
<th>Using Methods</th>
<th>Visual Hull Generation</th>
<th>Feature Extraction</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thinning based Skeletonization</td>
<td>370 ms</td>
<td>107 ms</td>
</tr>
<tr>
<td>3D bin distribution</td>
<td>370 ms</td>
<td>10 ms</td>
</tr>
<tr>
<td>Proposed method</td>
<td>3 ms</td>
<td>3 ms</td>
</tr>
</tbody>
</table>

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. 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 14 or 15 image sets per second. Therefore this method is matched with real-time recognition system. Fig. 10 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 ten 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, scaling and rotation, we can 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)).

V. CONCLUSION

In this paper, we proposed a 3D feature extraction method. 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 enhanced HAVH algorithm.
that the unnecessary processes are disabled such as the light and texture calculation. Therefore the proposed method can be applied to real-time recognition system. However, some problems remain in this method: error in visual hull rendering, limitation of the number of cameras, data transferring time in memories and distance calculation between overlapping of 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 of time and more accuracy to provide good performance.

REFERENCES


AUTHOR

Mr. M. Ramasubramanian received his B.Sc., and M.C.A Degrees in Bharathidasan University in the year 1997 and 2000 respectively, and received M.E. degree in the field of Computer Science and Engineering from Vinayaka Missions University in 2009. He is presently working as a Senior Assistant Professor, in Aarupadai Veedu Institute of Technology, Vinayaka Missions University, India. He is doing his Research in the area of Image Processing in the same University, under the guidance of Dr. M.A. Dorai Rangaswamy.