Title:
Fast and Accurate Triangular Model Generation for Shape from Silhouette

Authors:
Watchama Phothong, p_watchama@hotmail.com, National Central University, Jhongli, Taiwan
Tsung-Chien Wu, rabbit94577@gmail.com, National Central University, Jhongli, Taiwan
Jiing-Yih Lai, jylai@ncu.edu.tw, National Central University, Jhongli, Taiwan
Douglas W. Wang, dwmwang@gmail.com, Ortery Technologies, Inc.
Chao-Yang Liao, cyliao@ncu.edu.tw, National Central University, Jhongli, Taiwan
Ju-Yi Lee, juyilee@ncu.edu.tw, National Central University, Jhongli, Taiwan

Keywords:
Shape-from-Silhouette, Visual Hull, Mesh Generation, Octree Method, Model Reconstruction

DOI: 10.14733/cadconfP.2016.105-111

Introduction:
Most product presentations in e-commerce use various two-dimensional (2D) images of an object, mainly because these images are easy to process. However, 2D images provide only limited views of the object. Three-dimensional (3D) visualization is another technique for product presentation, in which multiple 2D images showing different views are integrated. The user can drag a point on the screen to orient a 2D image at a given angle. However, in such a representation, the information stored is highly redundant and the user may not be able to view the object from all desired angles. In addition, the real 3D shape and dimensions of the object cannot be obtained using this representation. Three-dimensional modeling with color texture is another method for product presentation in e-commence. Various techniques can be employed to create a 3D model—composed of triangular meshes—of an object.

The shape-from-silhouette (SFS) [1, 2, 6] method estimates the shape of an object from images of its silhouette. A typical SFS process includes the acquisition and processing of digital 2D images. The acquisition of 2D images depends on the devices used to capture images of the object from different angles. The process necessitates camera calibration because the object must be rotated to obtain images from different angles. In addition, these images must be captured sequentially. The associated software processing typically involves extracting silhouettes from 2D images, computing 3D points of the object, generating the triangular model from said 3D points, and texture mapping. The SFS method provides better estimates of object shape because it uses the boundary profile of the object from multiple 2D views. It relies on different algorithms for fast and accurate evaluation of 3D points from 2D silhouette points. However, surface cavities cannot be represented using the SFS method because silhouettes do not provide the required data [9]. Nevertheless, the SFS method is an effective method for estimating the 3D shape of an object and is considered useful in e-commerce applications.

Sablatnig et al. [7] proposed a volumetric method–based approach to reduce the time required to build the model as well as to reduce the number of views while ensuring a certain level of model accuracy by using octree of volume. They presented an algorithm for next-view planning with a minimal number of different views. Yemez et al. [10] presented a complete procedure that covered all aspects of the triangular model, from camera calibration to generation. They proposed a scheme combining octree construction and the marching cubes algorithm for generating the triangular model. Moreover, they developed an interpolation algorithm for accurately evaluating points on the marching cubes. Miline [4] developed a modified marching cubes method that can quickly compute an object's volume from its visual hull by using multiple views of the object. In this method, the first step is the voxelization of the

volume containing the target object. Then, the marching cubes algorithm is used to approximate the surface passing through the exterior voxels. Finally, positional accuracy of the surface is improved using binary search. Miline claimed that by applying the marching cubes algorithm to a low-resolution voxel-based model (as opposed to a high-resolution model), better results could be achieved in a shorter time. The binary search can further improve the marching cubes model at minimal computational expense.

Furthermore, SFS-based 3D reconstruction can be used to generate a model by employing exact polyhedra methods. Matusik et al. [3] computed an exact polyhedral representation of the visual hull directly from silhouettes. This method is well suited for rendering with graphic hardware, and it can be executed quickly because computations are performed during creation of the visual hull. Niem [5, 6] proposed a method for fast traversal of the layers of projected cones and retrieved the viewing edges lying on the surface of the visual hull, which amounts to a real-time full reconstruction model.

In this study, we developed an SFS method for generating triangular models of objects. The proposed algorithm is based on the direct intersection of multiple sets of infinite polygons from the silhouettes of 2D images to acquire the surface points of an object. The surface points are then triangulated according to the topological relationship of the obtained points and the polygons corresponding to each of the points. The proposed 3D modeling scheme is targeted at e-commerce applications, where the feasibility of real-time operation on a website is the primary concern. Therefore, we specify the following three quality indices to evaluate its performance: computational efficiency, number of vertices on the model, and model accuracy. The main contribution of the proposed method is that it can be used to compute the intersection points of infinite polygons directly and can hence yield the most accurate triangular model based on silhouette points of the object. To verify the performance of the proposed method, we compared it with another common method based on marching cubes [4, 10]. Finally, we present several realistic examples to demonstrate the feasibility of the proposed method.

Main Idea:
For generating 3D models from multiple images, the images should be captured in a controlled environment. 3D models are generated by capturing the bounding shape of the target objects from different angles obtained by varying camera rotation and orientation. Three-dimensional models can be constructed using many methods such as SFS and marching cubes. These methods are all based on visual hull computation, which involves creating a 3D model in accordance with the boundary profiles of a series of 2D images of an object. This study focuses on the generation of 3D points and triangular meshes from a series of silhouette points by using the octree reconstruction method.

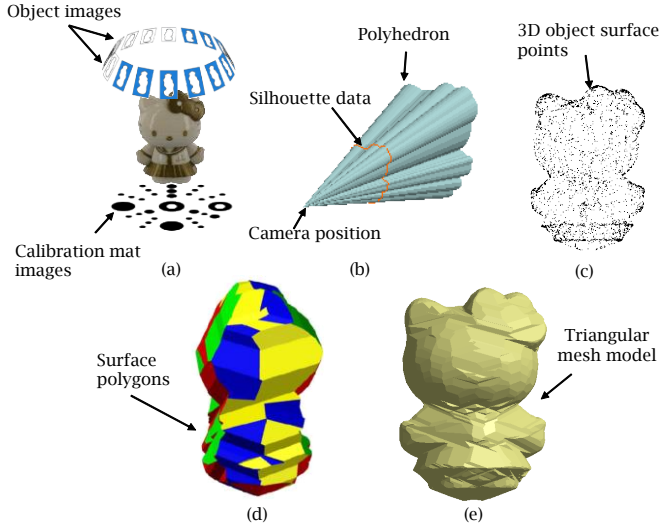

Fig. 1:An example demonstrating the integrated procedure of the proposed camera calibration, points generation and mesh generation algorithm, (a) input object images and calibration mat images, (b) polyhedron representing the outline shape on each image, (c) 3D object surface points, (d) surface polygons, and (e) triangular model

Figure 1 shows the overall flowchart of the proposed algorithm for generating 3D triangular meshes from 2D silhouette points. The first step in the proposed algorithm involves inputting silhouette points of all 2D images and the camera calibration data. Thereafter, the first volume that encloses the target object is computed. Subsequently, 3D polyhedra representing the outline shape of the object in all images are generated based on the silhouette points, camera point, and perspective projection property of the camera. A 3D point can be obtained by the intersection of three polygons. An octree construction algorithm is developed for

subdividing the first volume repeatedly until all small volumes having only three polygons intersected each other. The 3D points are then re-computed under additional conditions to eliminate redundant points. The relationships of all 3D points and those of the polygons with respect to each of the 3D points are recorded. This includes indices of polyhedral surfaces for each 3D point and point indices for each polygon. The aforementioned information is used to connect 3D points from a specific polygon to form a closed-loop polygon and to subsequently tessellate the polygons to form a polygon model. Finally, the polygon model is converted into triangular meshes and output as an STL file. The techniques used to achieve the aforementioned tasks are described below.

1. 3D point calculation

For each set of silhouette points on an image, a polyhedron must be generated. A polyhedron is a cone of the projected area from the camera point through all silhouette points. As shown in Fig. 2(a), the connection between the camera point and a silhouette point can form an edge, and the area between two neighboring edges can form an infinite polygon. If the edge length is restricted, a polygon is formed. Therefore, the polyhedron actually contains many polygons, and each polygon must pass through three specified points, namely, the camera point and two neighboring silhouette points, as shown in Fig. 2(a). A point on a 2D image actually indicates two neighboring polygons because each silhouette point can generate two polygons (Fig. 2(b)). Similarly, a line on a 2D image indicates one polygon because a line connects two neighboring silhouette points (Fig. 2(b)). The data to be recorded are (1) data related to each silhouette point, including two neighboring polygon indices and a view index, and (2) data related to each silhouette line, including a polygon index and a view index.

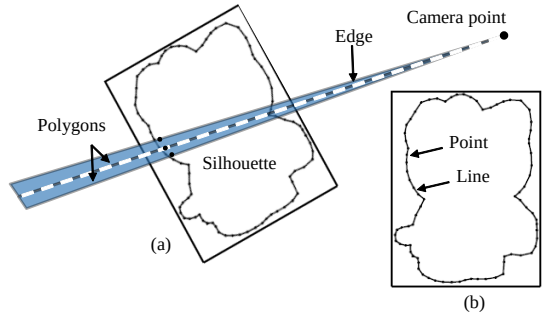

Figure 2: Terms used in this study, (a) a polygon is generated by camera point and two neighboring silhouette points, (b) point and line on the 2D image represent edge and polygon, respectively, on the 3D domain

1.1 Octree construction

An octree construction is established for determining the volumes having only three polygons intersecting with each other. Because many polygons are generated from multiple sets of silhouette points, octree construction is effective for subdividing the volume repeatedly and providing an effective parent–child relationship for all volumes created.

Owing to the use of octree construction, each volume is checked against every image, point, and line (see Fig. 2 for the definition of point and line) on each 2D image. Figure 3 shows the method of projecting a volume and checking its state. The volume is projected onto each image plane. Subsequently, the number of points that intersect or lie inside the projected volume is determined. Conditions for setting the state of the volume are as follows. After checking against all images, if the total number of polygons in a volume is greater than three, the state of that volume is “Subdivide volume”; if the number of polygons is equal to three, the state is “Calculate point”; if the number of polygons is less than three, the state is “Discard volume.”

To reduce the 3D point computation time, the images checked for points and lines on child volumes are deleted if no point and line lies inside an image and no line intersects with a projected volume of the parent volume. For

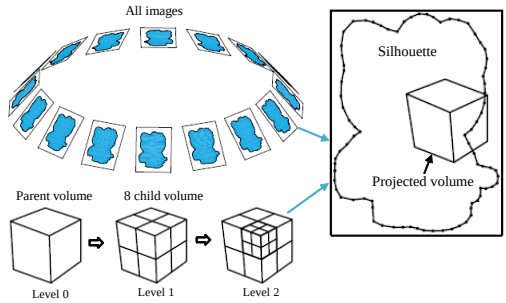

Fig. 3: Volume subdivision in accordance with the octree structure and the projection of each volume onto each image plane to check the status of the volume.

example, in Fig. 3, among the 16 experimental images, only five have points or lines inside the projected volume. Thereafter, this volume is subdivided into eight child volumes and all child volumes are projected onto these five images for checking, while the other 11 images are skipped.

2. Generation of Triangular Meshes

Once a set of 3D points generated from the intersection of the polyhedra is obtained, we can record the polygons that constitute each 3D point. In this step, each polygon is assigned a unique index, and the polygon indices corresponding to each point are recorded. The data structure of this record is shown in Fig. 4(a). Each polygon is essentially a plane and its boundary is formed by a series of 3D points. The data shown in Fig. 4(a) is employed to create a series of point indices corresponding to each polygon index, as shown in Fig. 4(b). The number of points constituting each polygon can be varied, and the sequence of points in each row is irregular.

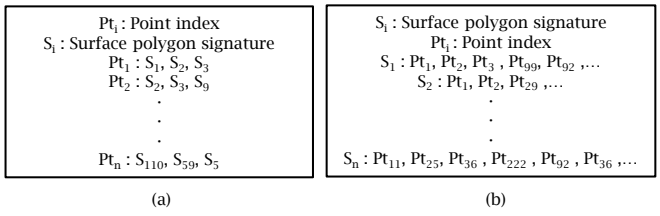

Fig. 4: Data structure of recording surface polygon and points indices, (a) surface polygon index, (b) points index.

A set of points located on the same polygon, as in Fig. 4(b), is arranged sequentially so that the points can be connected to form a polygonal mesh. Let the data in Fig. 4(a) be the polygon-index group and the data in Fig. 4(b) be the point-index group. Consider Fig. 5 as an example. The first polygon S_1 is chosen as the working polygon W_{poly} and all points corresponding to it are input from the point-index group. The first point Pt_1 is selected as the working point W_{point} and all polygons corresponding to it are input from the polygon-index group. Then, we choose one polygon N_{poly}, with the exception of W_{poly}, as the next polygon to test.

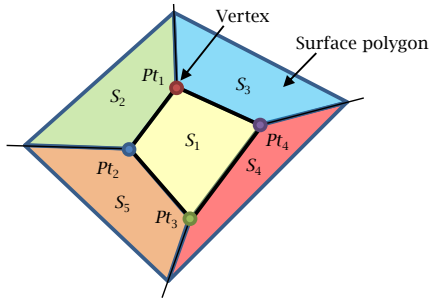

Fig. 5: The connection of surface polygons and points

Using W_{poly} and N_{poly}, the next point for generating the polygonal mesh is determined. To this end, we check the polygon-index group and find points having W_{poly} and N_{poly} simultaneously. The current W_{point} is placed in a polygon set $Poly_{mexh}$, and the point neighboring both W_{poly} and N_{poly} is set as the new W_{point}. With the new W_{point} and N_{poly}, the process is continued to find the next point in the sequence. Eventually, if the number of points is zero, the point-searching procedure is said to have ended and a series of $Poly_{mesh}$ having its connection order and representing the mesh model surface is obtained.

The $Poly_{mesh}$ series generated in the aforementioned process is collated and set as $Poly_{mesh\text{-}group}$. The last step is converting the obtained surface polygons into triangular meshes. Visualization Toolkit (VTK) [8] is employed to convert the polygons into triangular meshes. The data recorded in $Poly_{mesh}$ is used as the input for the VTK function, which outputs a set of triangular meshes converted from $Poly_{mesh}$. Finally, all triangular meshes are integrated to represent the object, and the result is saved as an STL file.

3. Examples and discussion

We employed several examples to evaluate the proposed method. The inputs were silhouette points extracted from multiple images and distributed uniformly around (360°) the object and the output was the surface triangular model of an object. In addition, we compared the results obtained using the proposed method with those obtained using the marching cubes method. Moreover, we compared the CPU times required for generating 3D points from silhouette points by using both methods. The simulations were performed on a personal computer with a 1.90 GHz CPU and 4 GB of RAM.

Figure 6 shows the original images and the corresponding reconstructed triangular models for six examples, where the left and right images in each figure panel denote the original image and the triangular model, respectively. Because the proposed method is based on visual hull computation, we focus on the accuracy of outline appearance. It is normal that any concavity on a part surface may not truly be reproduced with this method alone. Nevertheless, this model can be combined with texture mapping to provide a visually pleasing model for use in e-commerce applications. Remarkably, the proposed method truly represents spikes on an object, as shown in cases (d)–(f) in Figure 6. Table 1 lists the image parameters, number of vertices and faces on triangular models, and the CPU time required for two different stages, for the six examples in Fig. 6. According to Table 1, the number of faces in all models was less than 10,000, which will not induce any sluggishness during data download and website operation. The time required for "triangulation" was almost negligible compared to that of "3D point generation." The CPU time for all cases was between 41–132 s, indicating that the proposed method is acceptable in terms of computational efficiency.

Fig. 6: Original images and triangular meshes reconstructed for six examples, (a) cup, (b) owl, (c) cat doll, (d) vase, (e) Gundam, and (f) horse.

Case	Images		Triangular meshes		CPU time (sec)	
	No. views	Silhouette points/view	Vertices	Faces	3D points generation	Triangulation
Cup	16	100	3802	7600	132	0.618
Owl	16	100	3640	7276	77	0.509
Cat doll	16	100	3416	6828	87	0.464
Vase	16	100	4672	9332	120	0.935
Gundam	16	100	4482	8948	42	0.572
Horse	16	100	6796	13592	207	1.279

Table 1: Parameters on images, triangular meshes obtained and CPU times required for six examples

For performance verification, the results obtained using the proposed method were compared to those obtained using the marching cubes method [7, 10]. In the marching cubes method, a level R is assigned to determine the number of layers into which the first volume is subdivided. An octree structure is established to subdivide the volume into pieces of the same size based on level R.

Case	No. points/ image	No. views	Triangular meshes		CPU time (sec)
			Vertices	Faces	
Proposed method	200	16	6796	13592	207
R=5	200	16	3654	7296	14
R=6	200	16	15057	30048	60
R=7	200	16	62184	123958	257

Table 2: Parameters on images, triangular meshes obtained and CPU time for the proposed method and those using marching cubes method

Figure 7 shows the case "horse," which was employed to evaluate the performance of both methods. The first two plots in the first row (Figs. 7(a) and (b)) are the triangular models generated at R= 5 and 6, respectively. These two models were unsatisfactory because they appear to be deviated from the original object shape. The three plots in the second row are models obtained using the marching cubes

method with R = 7, that obtained using the proposed method, and a comparison of both models. The model obtained using the marching cubes method with R = 7 is quite similar to that obtained using the proposed method. However, considerable noise was observed on the model obtained using the marching cubes method because the vertices were interpolated. Every vertex in the model obtained using the proposed method represents the intersection of three polygon faces. Table 2 lists the vertices and faces of the triangular models, as well as CPU times required for the aforementioned five cases. A drawback of the marching cubes method is that the number of vertices and faces, and CPU time are very high at R = 7, which makes it unacceptable for real-time operation on a website. Therefore, we conclude that the proposed method outperforms the marching cubes method in terms of accuracy, model complexity, and the overall computational time required.

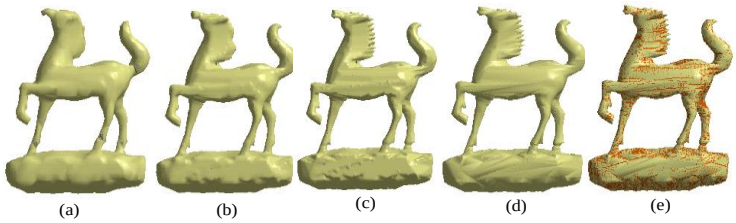

Fig. 7:Triangular meshes generated, (a) R=5, (2) R=6, (c) R=7, (d) proposed method, and (e) comparison of (c) and (d)

Conclusion:

In this study, we proposed a method for calculating 3D points and triangular meshes based on the SFS technique and visual hull intersection from multiple object images captured in a controlled imaging environment. The proposed algorithm employs an octree structure for efficient intersection check and hence for obtaining 3D points. In addition, we developed an improved algorithm to reduce the overall computation time. This method performed very well and generated 3D points accurately and quickly compared to the marching cubes method. However, because the 3D model was created using SFS, it cannot generate the shape of any concavity on an object because SFS considers only the bounding of a silhouette, which does not provide details about any concave area on the object. A texture mapping technique could be developed to add texture on the model in order to make it visually pleasing, as is desired in e-commerce applications.

References:

[1] Brian, S. M.; Curlee, B.; Diebel, J.; Scharstein, D.; Szeliski, R.: A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms, Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1, 2006, 519-528. http://dx.doi.org/10.1109/CVPR.2006.19

[2] Chen, Q.; Medioni, G.: A Volumetric Stereo Matching Method Application to Image-Based Modeling, Computer vision and pattern recognition, 2006 IEEE Computer Society Conference, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 1, 1999, 29-34. http://dx.doi.org/10.1109/CVPR.1999.786913

[3] Matusik, W.; Buehler, C.; Mcmillan, L.: Polyhedral visual hulls for real-time rendering, Rendering Techniques Part of the Series Eurographics, 2001, 115-125. http://dx.doi.org/10.1007/978-3-7091-6242-2_11

[4] Milne, P. S.: Visual Hulls for Volume Estimation: A Fast Marching Cubes Based Approach, M. S. Thesis, University of Cape Town, 2005

[5] Niem, W.; Buschmann, R.: Automatic Modelling of 3D Natural Objects from Multiple Views, Image Processing for Broadcast and Video Production, 1995, 181-193. http://dx.doi.org/10.1007/978-1-4471-3035-2_15

[6] Niem, W.: Robust and Fast Modelling of 3D Natural Objects from Multiple Views, Image and Video Processing II, 2182, 1994, 388–397. http://dx.doi.org/10.1117/12.171088

[7] Sablatnig, R.; Tosovic S.; Kampel, M.: Next view planning for shape from silhouette, Proceedings of the 8th Computer Vision Winter Workshop, 2003, 77-82.
[8] Schroeder, W. J.; Martin, K. M.: The Visualization Toolkit, 2005, 593 – LV. http://dx.doi.org/10.1016/B978-012387582-2/50032-0
[9] Yemez, Y.; Sahilioglu, Y.: Shape from silhouettte using topology-adaptive mesh deformation, Pattern Recognition Letters, 30(13), 2009, 1198-1207. http://dx.doi.org/10.1016/j.patrec.2009.05.012
[10] Yemez, Y.; Schmitt, F.: 3D reconstruction of real objects with high resolution shape and texture, Image and Vision Computing, 22(13), 2004, 1137–1153. http://dx.doi.org/10.1016/j.imavis.2004.06.001