COMPUTER-AIDED DESIGN & APPLICATIONS, 2017
VOL. 14, NO. 4, 436–449
http://dx.doi.org/10.1080/16864360.2016.1257186

Fast and accurate triangular model generation for the shape-from-silhouette technique

Watchama Phothong [a], Tsung-Chien Wu [a], Jiing-Yih Lai [a], Douglas W. Wang [b], Chao-Yaug Liao [a] and Ju-Yi Lee [a]

[a]National Central University, Taoyuan, Taiwan; [b]Ortery Technologies, Inc., Taiwan

ABSTRACT

In this study, we propose a method for generating triangular meshes of objects based on the shape-from-silhouette (SFS) technique. The proposed algorithm uses the direct intersection of multiple sets of infinite polygons from the silhouettes of two-dimensional (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. In addition, a comprehensive study between the proposed method and the marching-cubes method was conducted. The main advantages of the proposed method are that all 3D points are precisely located on the silhouettes of 2D images, and that the number of vertices on the triangular model is very efficient compared to that for the results from the marching-cubes method. In contrast, the marching-cubes method faces noise and resolution problems, where the former is because the vertices are interpolated, and the latter is because the maximum level is restricted. Several realistic examples are presented to demonstrate the feasibility of the proposed method.

KEYWORDS

Mesh generation; Model Reconstruction; Octree method; Shape-from-silhouette; Visual hull

1. Introduction

Most product presentations in e-commerce use two-dimensional (2D) images of an object, mainly because these images are easy to process. However, 2D images provide only limited views of an 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 actual 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-commerce. Various techniques can be employed to create a 3D model, composed of triangular meshes, of an object.

The shape-from-silhouette (SFS) [2, 3, 8] 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 a triangular model from said 3D points, and texture mapping. The SFS method provides good 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 [14]. Nevertheless, the SFS method is an effective method for estimating the 3D shape of an object and is considered to be useful in e-commerce applications.

Sablatnig et al. [11] proposed a volumetric method–based approach to reduce the time required to build a model and to reduce the number of views while ensuring a certain level of model accuracy by using an octree of a volume. They presented an algorithm for next-view planning with a minimal number of different views. Yemez et al. [15] presented a complete procedure that covered all aspects of the triangular model, from camera cali-

CONTACT Watchama Phothong p_watchama@hotmail.com

bration to generation. They proposed a scheme combining octree construction and the marching-cubes (MC) algorithm for generating a triangular model. Furthermore, they developed an interpolation algorithm for accurately evaluating points on the marching cubes. Miline [6] 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, the positional accuracy of the surface is improved using a 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 with minimal computational expense.

Furthermore, SFS-based 3D reconstruction can be used to generate a model by employing exact polyhedral methods. Matusik et al. [5] 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 the creation of the visual hull. Niem [8, 9] 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 uses 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 specified 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 directly compute the intersection points of infinite polygons and can, hence, yield the most accurate triangular model based on the silhouette points of the object. To verify the performance of the proposed method, we compared it with another common method based on marching cubes [6, 15]. We will also present several realistic examples to demonstrate the feasibility of the proposed method.

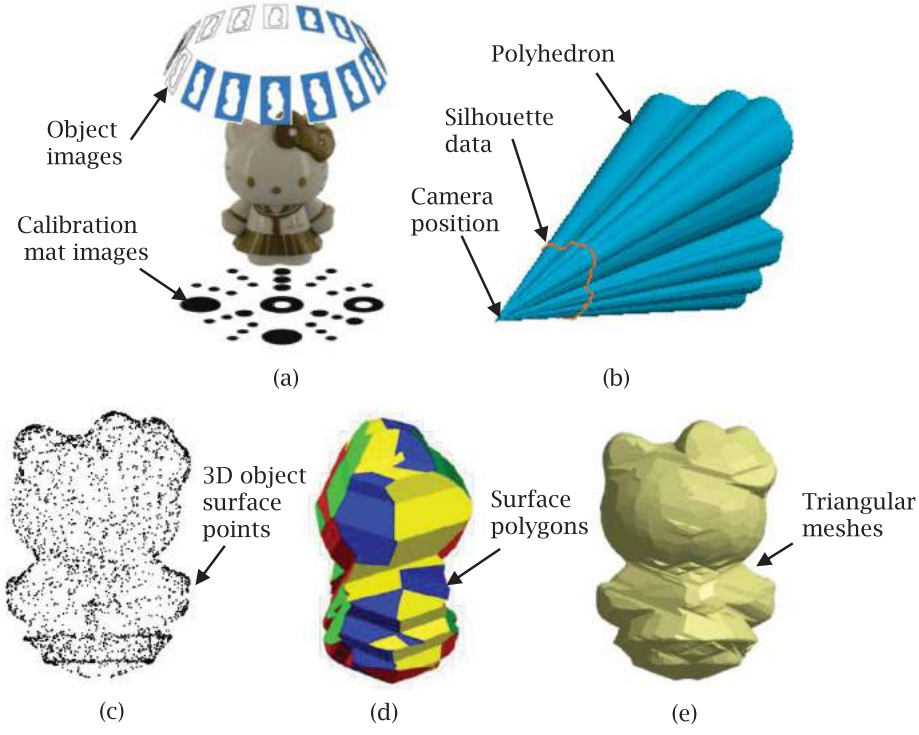

Figure 1. An example demonstrating the integrated procedure of the proposed camera calibration, points generation, and mesh generation algorithm. (a) Input object images and calibration mat images, (b) a polyhedron representing the outline shape of each image, (c) 3D object surface points, (d) surface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

In order to generate 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 the 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 focused on the generation of 3D points and triangular meshes from a series of silhouette points using the octree reconstruction method.

Figure 1 shows an 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. Then, 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 from the intersection of three polygons. An octree construction algorithm was developed for subdividing the first volume repeatedly until all small volumes have only three polygons intersecting with 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. This 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 these tasks are described below.

2. 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 finite polygon is formed. Therefore, the polyhedron actually contains many polygons, and each polygon must pass through three specific 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.

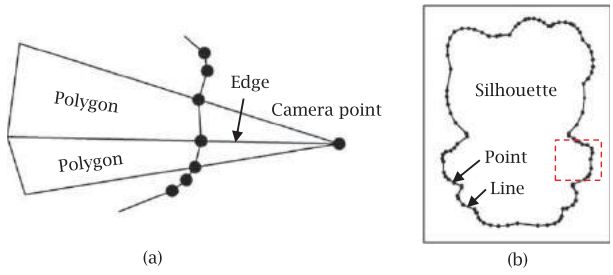

Figure 2. Terms used in this study: (a) a polygon is generated from a camera point and two neighboring silhouette points, (b) a point and line on the 2D image represent an edge and polygon, respectively, on the 3D domain.

The flowchart for calculating 3D points from multiple sets of silhouette points is shown in Fig. 3; it is composed of the following four steps: (1) the first volume that covers the entire object is defined, (2) octree construction that generates an eight-child volume for each parent volume is established, (3) the status of each test volume is determined, and (4) 3D points are computed from polygon intersections. A cube example is employed to explain the intermediate results of the procedures in Fig. 3. Four views of the cube are captured, as shown in Fig. 4(a). The corresponding four sets of silhouette points extracted from the above four images, respectively, are shown in Fig. 4(b).

2.1. Defining the first volume

The first volume that covers the entire object is computed as follows. A two-dimensional bounding box corresponding to the silhouette points on each image is computed. Then, a polyhedron is obtained by connecting the camera point and four boundary points of the bounding box. All polyhedra from different images are intersected to find a set of 3D intersection points. The bounding box of all intersection points can cover the object and is, hence, considered to be the first volume. Fig. 5(a) shows the polyhedra from four different views of images for the cube example. The bounding box of all intersection points denotes the first volume, as shown in Fig. 5(b).

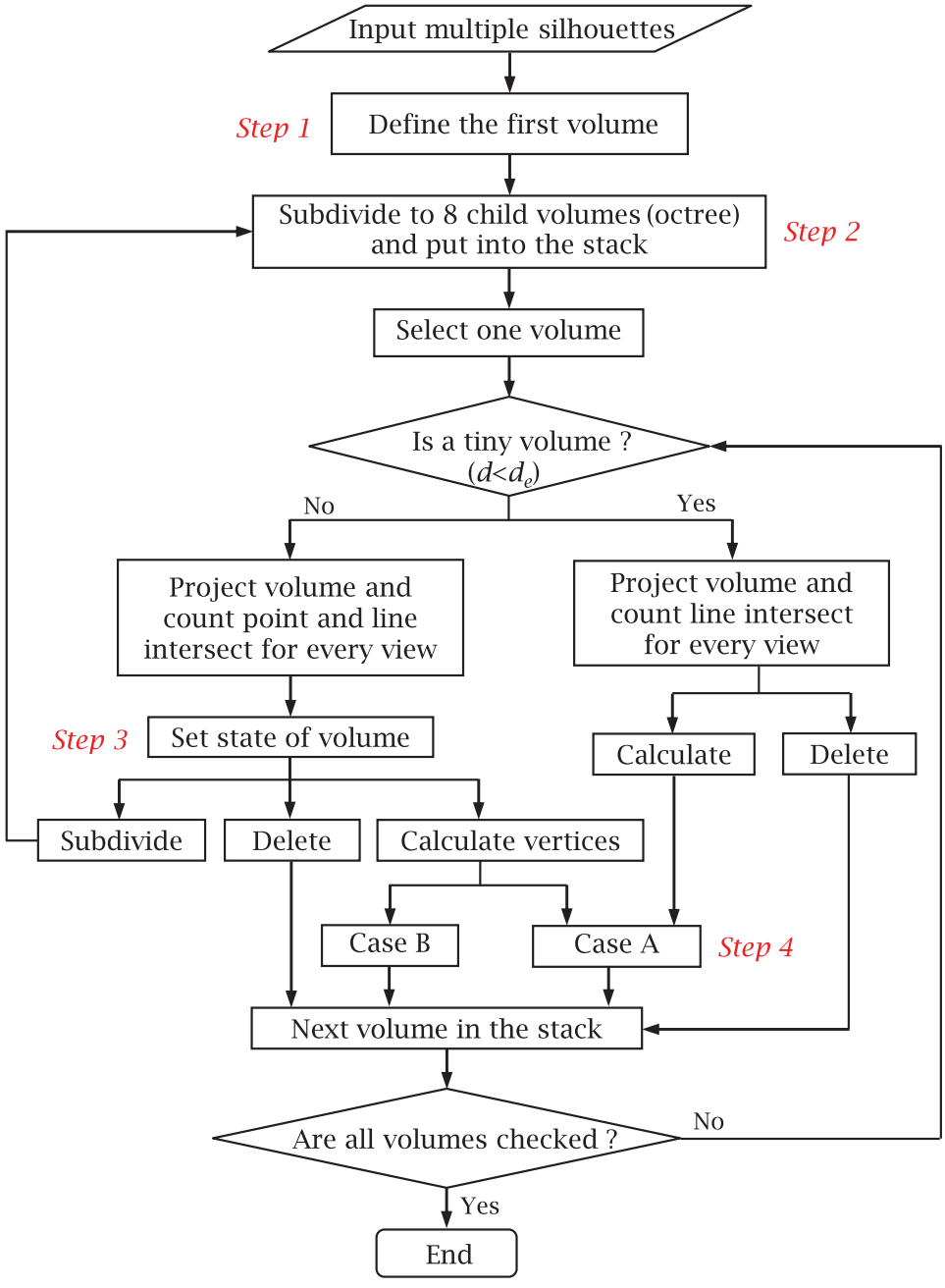

Figure 3. Flowchart for calculating 3D points from multiple sets of silhouette points.

2.2. Volume subdivision and state checking

An octree construction is established for determining the volumes that have only three polygons intersecting with each other. Since 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. As shown in Fig. 3, each parent volume is divided into eight child volumes. Each child volume is checked to determine whether it is formed by the intersection of more than three polygons. If it is, this volume is again subdivided to eight child volumes. This process is repeated until the volume is comprised of only three polygons or the length d of the volume is less than the allowable tolerance d_e. Fig. 6 depicts that the first volume of the cube example is subdivided into eight child volumes.

Owing to the use of octree construction, each volume is checked against every image, point, and line (see Fig. 2 for the definition of a point and a line) on each 2D image. Figure 7 shows the method for projecting a volume and checking its status. The volume is projected onto

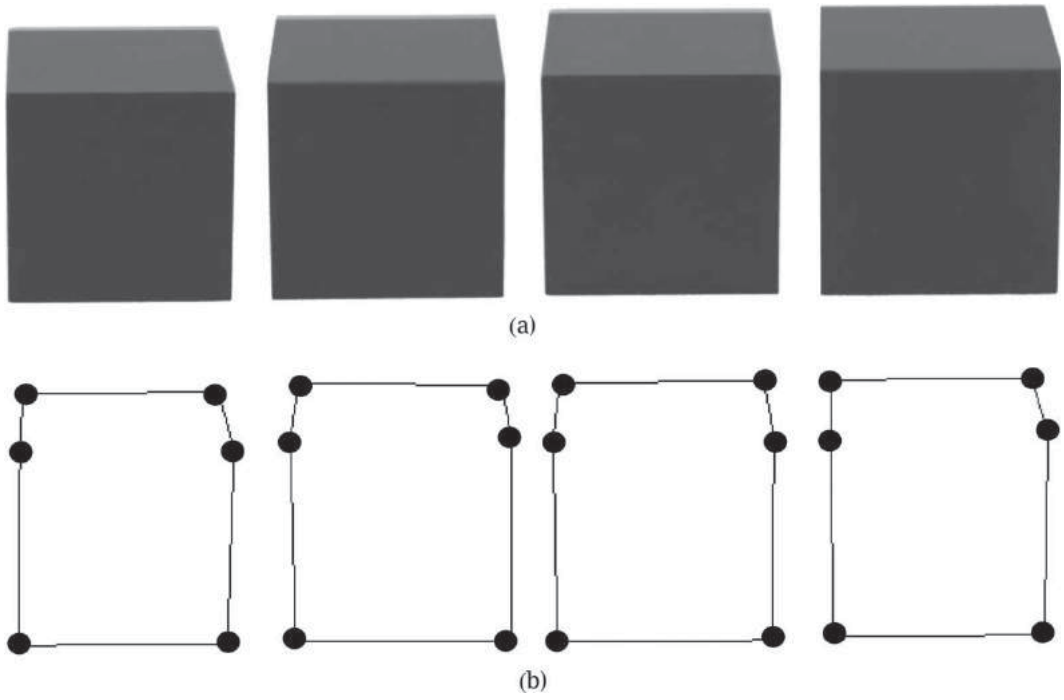

Figure 4. A cube example to illustrate the immediate results of the procedures in Fig. 3, (a) four images captured from different views, (b) four sets of silhouette points extracted from the above four images, respectively.

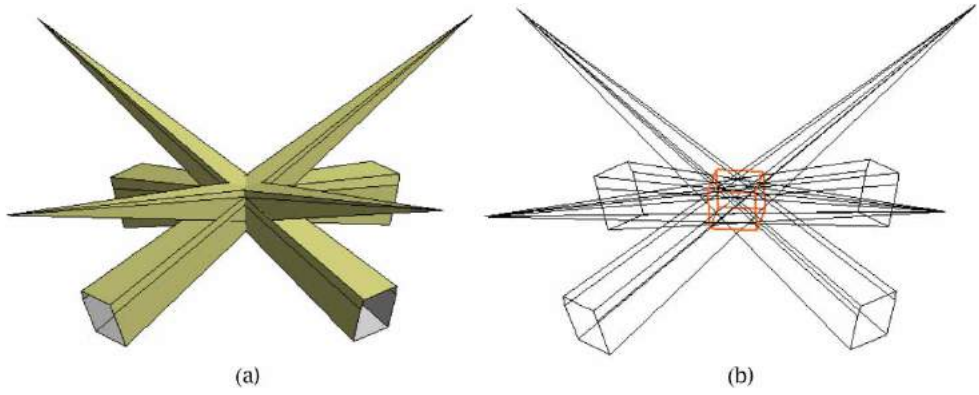

Figure 5. Defining the first volume for the cube example, (a) a polyhedra generated from four different views of images, (b) the first volume determined from the bounding box of all intersection points.

each image plane. Subsequently, the number of points that intersect or lie inside the projected volume is determined. The conditions for setting the status of the volume are as follows (Fig. 3). After checking against all images, if the total number of polygons in a volume is greater than three, the status of that volume is "Subdivide volume"; if the number of polygons is equal to three, the status is "Calculate point"; if the number of polygons is less than three, the status is "Discard volume."

2.3. Conditions for 3D point calculation

There are two conditions for computing 3D points from polygons. The first condition is Case A, in which a 3D point is intersected by three polygons originating from 0 points and 3 lines on 2D images. In this case, the three polygons are from three different images (Fig. 8(a)). The second condition is Case B, in which a 3D point is intersected by three polygons originating from 1 line of the first image and 1 point of the second image. In this case, two of the connected polygons are from the first image, while the third polygon is from the second image (Fig. 8(b)). For a tiny volume, only Case A is valid because the intersecting polygons come from three different images.

To reduce the 3D point computation time, the images checked for points and lines on child volumes are deleted if no point or line lies inside an image and no line

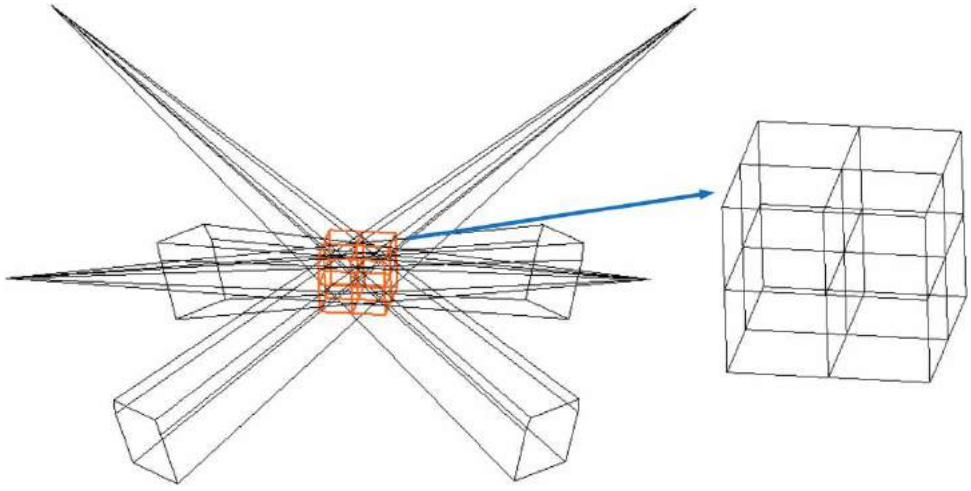

Figure 6. Eight child volumes obtained for the cube example using the octree construction.

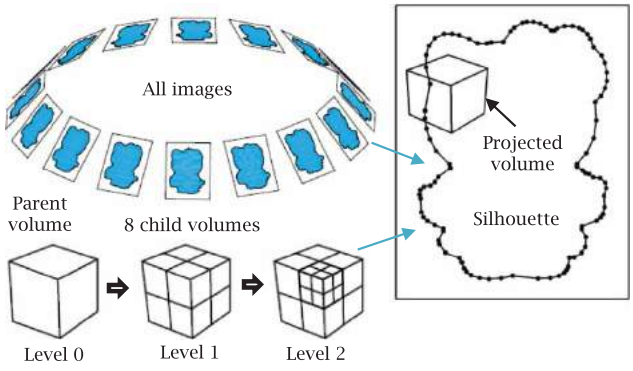

Figure 7. Volume subdivision in accordance with the octree structure and the projection of each volume onto each image plane to check the status of the volume.

intersects with a projected volume of the parent volume. For example, in Fig. 7, among the 16 experimental images, only five have points or lines inside the projected volume. As such, 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.

3. 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 stored of these records is shown in Fig. 9(a). Each polygon is essentially a plane and its boundary is formed by a series of 3D points. The data shown in Fig. 9(a) is employed to create a series of point indices corresponding to each polygon index, as shown in Fig. 9(b). The number of points constituting each polygon can be varied, and the sequence of points in each row is irregular.

Points located on the same polygon, as in Fig. 9(b), are arranged sequentially so that they can be connected to form a polygonal mesh. Let the data in Fig. 9(a) be the polygon-index group and the data in Fig. 9(b) be the point-index group. 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

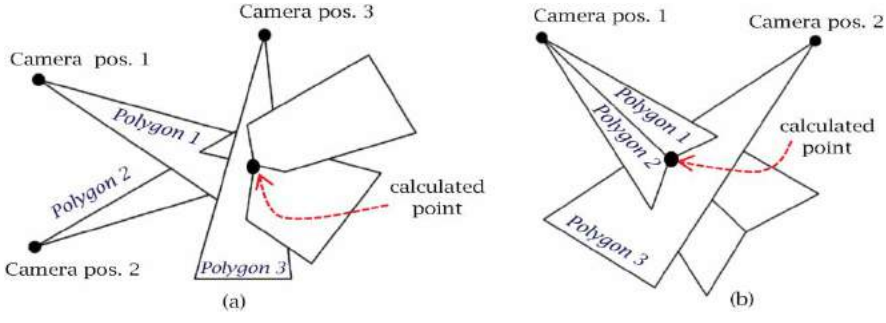

Figure 8. Calculated 3D point. (a) Case A, (b) Case B.

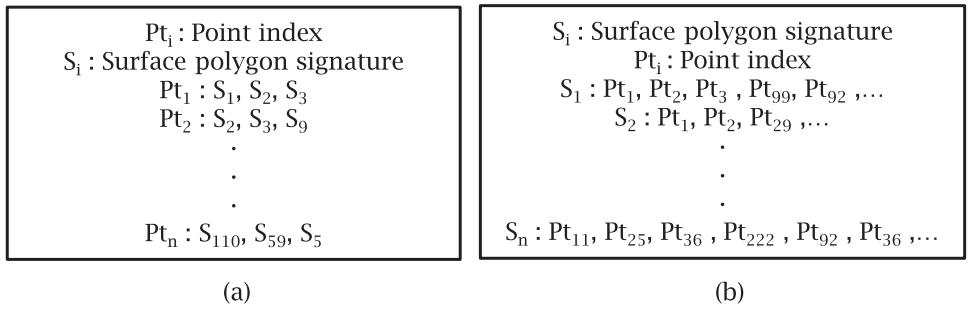

Figure 9. Data stored for recording surface polygon and point indices: (a) surface polygon index, (b) points index.

exception of W_{poly}, as the next polygon to test. Consider Fig. 10 as an example. Let S_1 be W_{poly}, and the corresponding point indices be Pt_1, Pt_2, Pt_3, and Pt_4. Given that Pt_1 is the first point, it is set as W_{point}. The corresponding polygons of Pt_1 are S_1, S_2, and S_3. Here S_2 is chosen as N_{poly} because it is next to S_1.

Using W_{poly} and N_{poly}, the next point for generating the polygonal mesh can be determined. To this end, we will check the polygon-index group and find points that have 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}. Additionally, the third neighboring polygon of the new W_{point} is set as N_{poly}. In Fig. 10, only Pt_2 neighbors both $S_1(W_{poly})$ and $S_2(N_{poly})$; hence, it satisfies the condition and is included in $Poly_{mexh}$ and Pt_2 is set as the new W_{point}. Furthermore, S_5 is set as N_{poly} because it is the third polygon of Pt_2 (the other two polygons S_1 and S_2 have been used already). 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 ends. The point-index group is checked to find points that have not been processed. If all points have been processed, the procedure is shifted to the next surface polygon to search for the sequence of points. This process is stopped after all surface polygons have been processed. Finally, a series of $Poly_{mesh}$ with connection order and representing the mesh model surface is obtained. Fig. 11 shows the result of the aforementioned method implemented on the cube example, where the points on each surface polygon have been arranged in a counterclockwise direction.

The $Poly_{mesh}$ series generated is collected and set as $Poly_{mesh-group}$. The last step is to convert the obtained surface polygons into triangular meshes. Since all points on a polygon set $Poly_{mesh}$ are co-planar, it is easy to generate a set of triangular meshes from $Poly_{mesh}$ When all $Poly_{mesh}$ on $Poly_{mesh-group}$ are triangulated, a triangular model of the object can be obtained. Finally, the result is saved as an STL file. Figure 12 shows the triangular meshes obtained for the cube example.

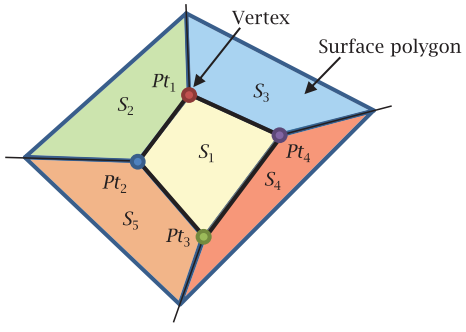

Figure 10. The connection between surface polygons and points.

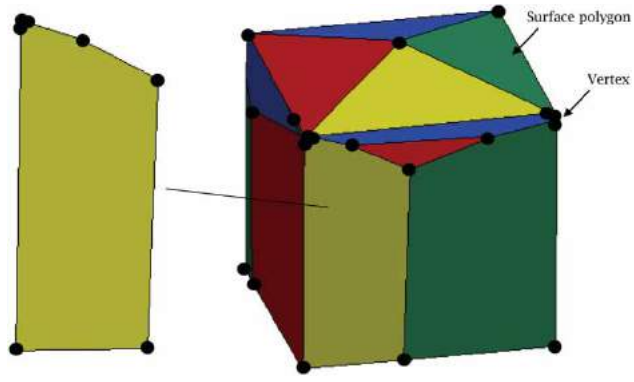

Figure 11. Surfce polygons obtained for the cube example.

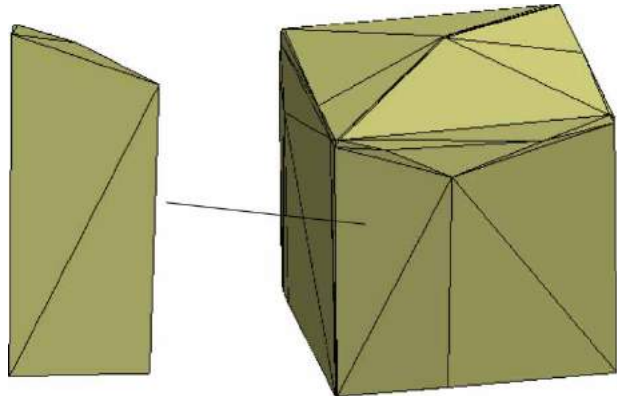

Figure 12. Triangular meshes obtained for the cube example.

4. 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. Furthermore, we compared the required CPU time for generating 3D points from silhouette points using both methods. The simulations were performed on a personal computer with a 1.90 GHz CPU and 4 GB of RAM.

Figure 13 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. Since the proposed method is based on visual hull computation, we focused on the accuracy of outline appearance. It is normal that any concavity on a partial surface may not truly be reproduced using this method alone. Nevertheless, this model can be combined with texture mapping to produce 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 Fig. 13. 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. 13. According to Table 1, the number of faces in all models was less than 15,000, which will not induce any sluggishness during data download and website operation. The time required for triangulation was almost negligible compared to the time required for 3D point generation. The required CPU time for all cases was between 42–207 s, indicating that the proposed method is acceptable in terms of computational efficiency.

For performance verification, the results obtained using the proposed method were compared to those obtained using the marching-cubes method [11, 15]. 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. Each subdivided volume is projected onto the 2D images

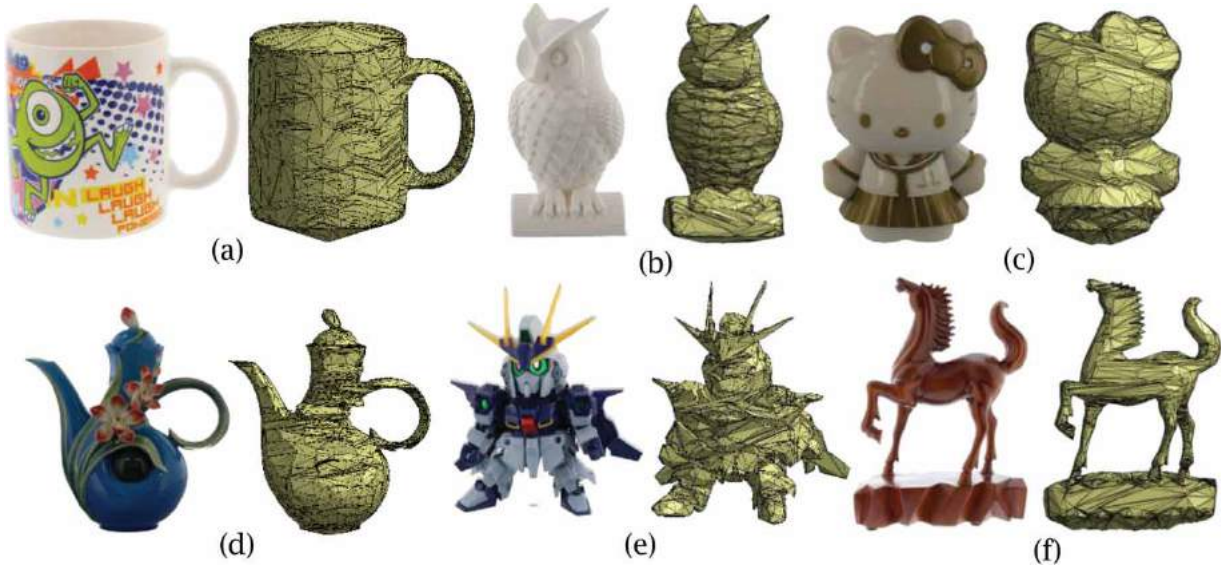

Figure 13. Original images and triangular meshes reconstructed for six examples: (a) cup, (b) owl, (c) cat doll, (d) teapot, (e) robot toy, (f) horse figurine.

Table 1. Parameters of images, obtained triangular meshes, and required CPU times for six examples.

	Images		Triangular meshes		CPU time (s)	
Case	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
Teapot	16	100	4672	9332	120	0.935
Robot toy	16	100	4482	8948	42	0.572
Horse	16	100	6796	13592	207	1.279

sequentially to determine whether it intersects the silhouette of the images. The volumes that intersect the silhouettes are retained, indicating that they are located on the object surface. The marching-cubes method, which incorporates an interpolation algorithm, is then employed to generate a triangular mesh for each volume lying on the object surface. The connection of all meshes represents the triangular model of the object. R affects the resolution of the volumes that can be subdivided. We conducted a series of studies to investigate the effect of R and compared the results obtained using the proposed method and the marching-cubes method.

Figure 14 shows the case "horse," which was employed to evaluate the performance of both methods. The first two plots (14(a) and (b)) are the triangular models generated for $R = 5$ and 6, respectively. These two models were unsatisfactory because they appear to deviate from the original object shape. The third and fourth plots (14(c) and (d)) are models obtained using the marching-cubes method with $R = 7$ and that obtained using the proposed method, respectively. The model obtained using the marching-cubes method with $R = 7$ is quite similar to that obtained using the proposed method. However, considerable noise can be 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 the required CPU times for the aforementioned four cases. A drawback of the marching-cubes method is that the number of vertices and faces, and required CPU time are very high for $R = 7$, which makes it unacceptable for real-time operation on a website.

An error analysis algorithm was employed to evaluate the errors between two triangular models. The model obtained using the proposed method served as a reference and the distance between each of the vertices on a model obtained using the marching-cubes method and the reference model was evaluated. The maximum (Max) and root-mean-square (RMS) distances for all vertices were evaluated, representing the Max and RMS errors, respectively, between two models. The errors for $R = 5$, 6, and 7 are also listed in Tab. 2. The results indicate that the marching-cubes method with $R = 7$ yields the minimum RMS error (0.631 mm here), indicating that the model with $R = 7$ is the most similar to that of the proposed method. However, the Max error is still up to 9.471 mm, which indicates that some of the points in the model with $R = 7$ deviated from the model obtained using the proposed method. Fig. 15 depicts the distribution of the errors for $R = 5$, 6, and 7, respectively. These plots clearly indicate that the model with $R = 7$ is the best because the error vectors are much smaller than those of the models with $R = 5$ and 6. Furthermore, considerable errors appear near the horse's mane because its shape is more irregular.

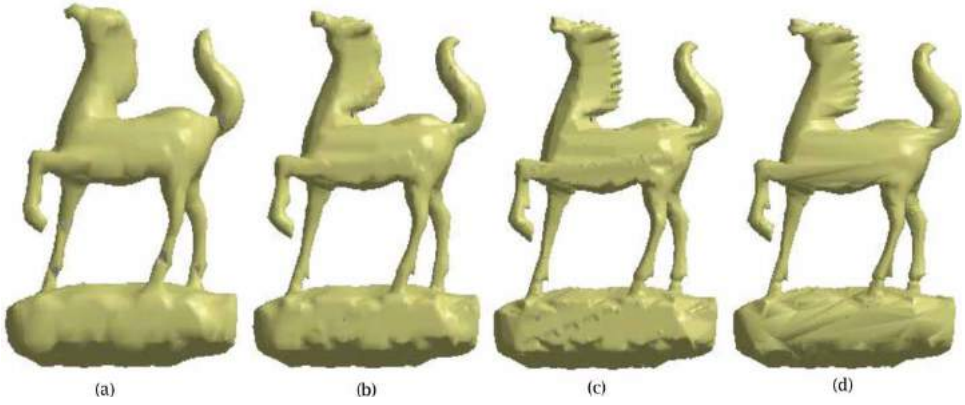

Figure 14. Generated triangular meshes: (a) MC with $R = 5$, (2) MC with $R = 6$, (c) MC with $R = 7$, (d) the proposed method.

Table 2. Parameters of images, obtained triangular meshes, and required CPU time for the proposed method and cases using marching-cubes method.

Case	No. points/ image	No. views	Triangular meshes		Error (mm)		CPU time(s)
			Vertices	Faces	Max	RMS.	
Proposed method	200	16	6796	13592	Reference		207
R = 5	200	16	3654	7296	14.152	1.930	14
R = 6	200	16	15057	30048	9.335	1.070	60
R = 7	200	16	62184	123958	9.471	0.631	257

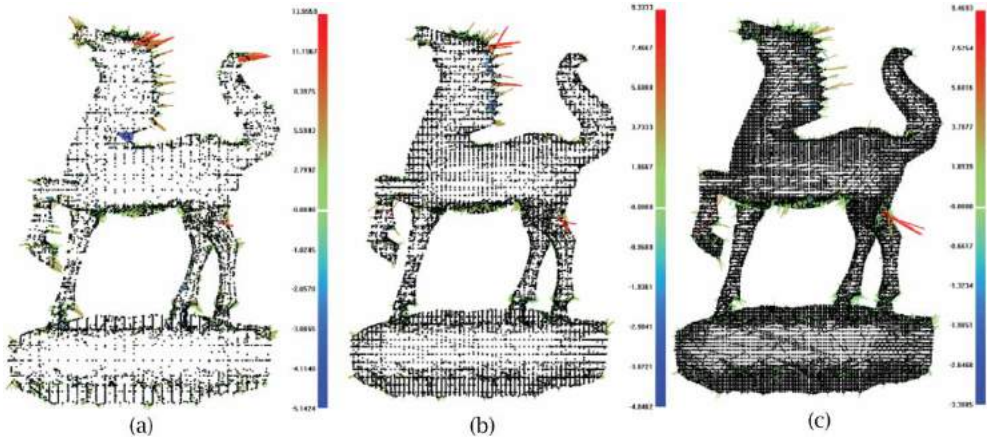

Figure 15. Error comparison between models obtained using the marching-cubes method and that obtained using the proposed method, (a) $R = 5$, (2) $R = 6$, (c) $R = 7$.

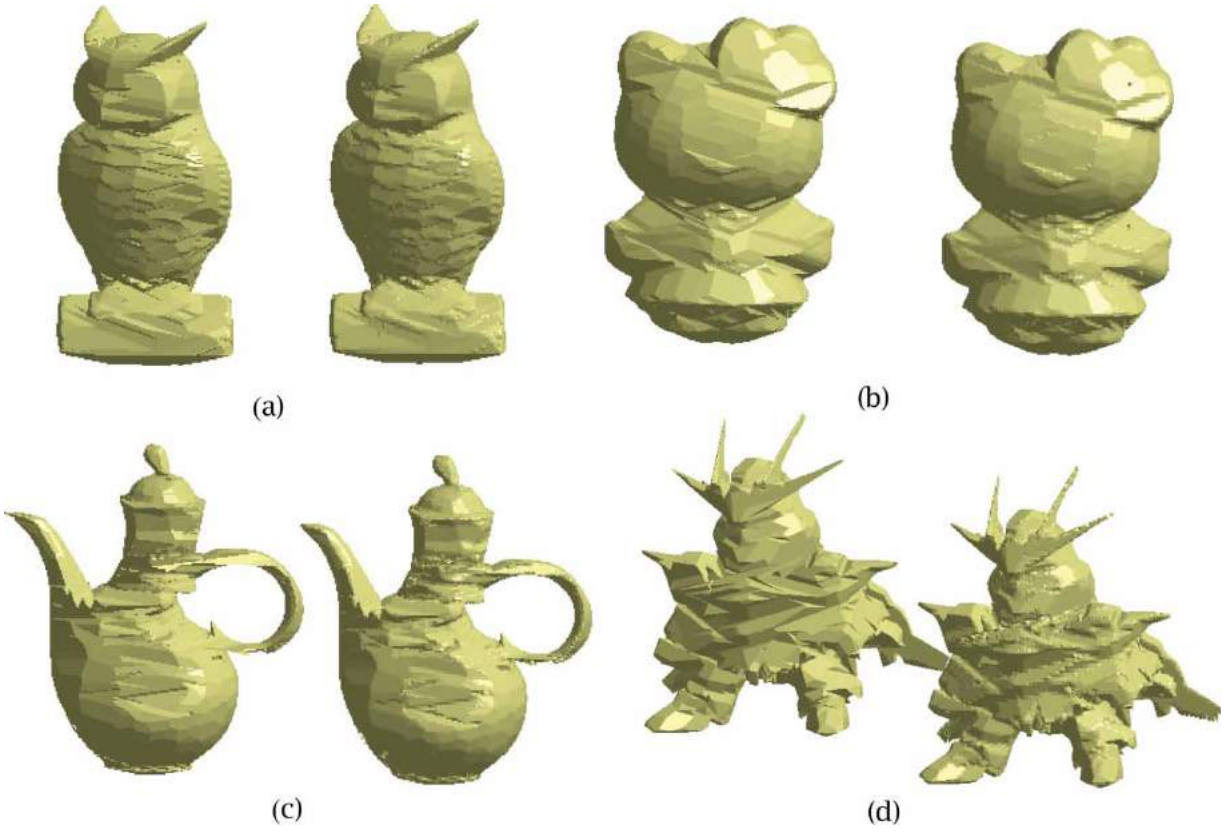

Figure 16. Comparison between triangular meshes of the proposed method (left portions) and the MC method (right portions): (a) owl, (b) cat doll, (c) teapot, (d) robot toy.

A detailed comparison between the model obtained using the marching-cubes method with $R = 7$ and that obtained using the proposed method is described below. Figure 16 shows the models for four examples, where the left and right models in each figure panel denote the proposed method and the marching-cubes method, respectively. The overall appearance of each pair of models is quite similar, but the number of vertices and triangular faces on each pair of models are different. Table 3 lists the number of vertices and faces of the triangular models, and the CPU time required for both methods. All examples show that the number of vertices and faces for the marching-cubes method are quite large because it calculates the vertices of every cube lying on the object surface.

Table 3. Parameters of the proposed method and marching cube intersections for five examples.

	Proposed method			Marching cube method		
Case	Vertices	Faces	CPU time(s)	Vertices	Faces	CPU time(s)
Cup	3802	7600	132	60174	120352	132
Owl	3640	7276	77	56334	112516	122
Cat doll	3416	6828	87	65190	130364	94
Teapot	4672	9332	120	68489	136966	134
Robot toy	4482	8948	42	54133	108244	131

However, for the proposed method, a vertex is computed only if three polygons from the polyhedrons intersect each other.

Another disadvantage of the marching-cubes method is that noise appears because the vertices are obtained from interpolation. As shown in Fig. 17, for the proposed method, all 3D points are located on the boundaries of polygonal meshes, and all points on a polygonal mesh are essentially co-planar. The triangulation of each polygonal mesh can yield a set of triangles all lying on the same plane. However, for the marching-cubes method, many small cubes of equal size are inserted uniformly on a polygonal mesh, which yields many triangular faces on the polygonal mesh. As all vertices are interpolated, some of the triangular faces may be across two different polygonal meshes, which results in noise, as shown on the middle plot in Fig. 17. For a model obtained using the visual hull technique, a large number of vertices and faces are unnecessary because no information inside each polygonal mesh is provided on the original images. Therefore, a model obtained using the proposed method is actually more accurate as a representation of the object shape.

For sharp surfaces, such as spikes on an object, the result obtained using the marching-cubes method tends to yield considerable noise, or even inaccurate shapes. Figure 18 shows that the spikes on the model obtained using the marching-cubes method are not only rough, but are also shorter than those obtained using the proposed method. This is because the cubes used in the marching-cubes method are not fine enough. Finer cubes can be obtained by increasing the level R. However, when R is increased, the required memory and computation time increase tremendously. This resolution problem is the primary limitation of the marching-cubes method. In fact, we tried to increase the level R to 8, but the number

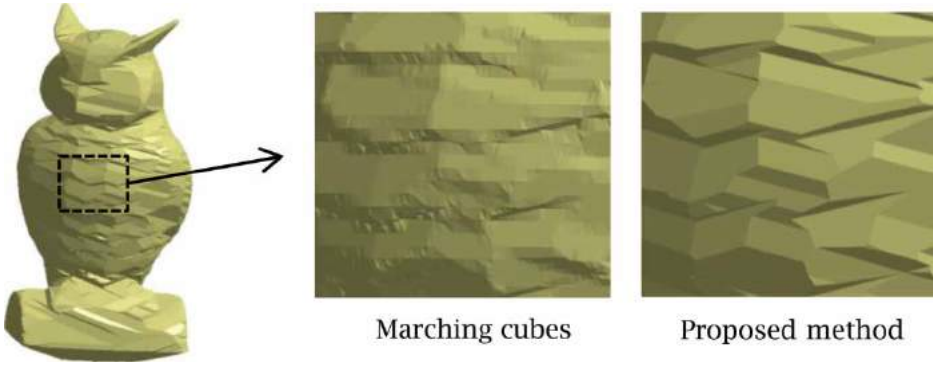

Figure 17. Noise problem of the marching-cubes method due to the interpolation of vertices.

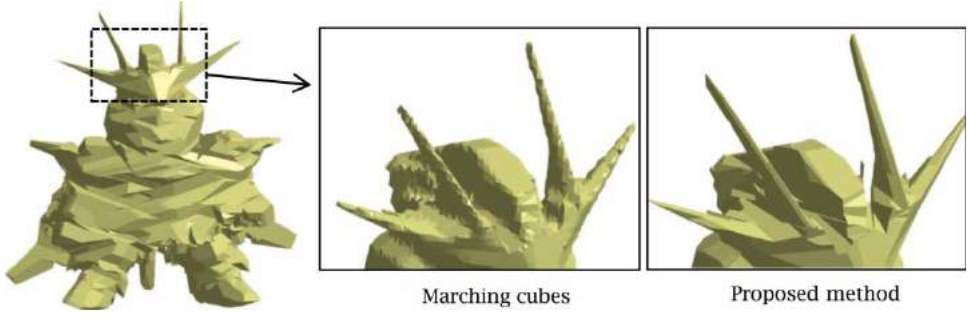

Figure 18. Resolution problem of the marching-cubes method due to the limitation of the level R.

of vertices on the triangular model and the computation time were increased to unreasonable values. In contrast, the result obtained using the proposed method does not have this problem, as shown on the right plot in Fig. 18.

For objects composed of fine and composite shapes, the proposed method tends to yield better results than the marching-cubes method. Figure 19 compares a local composite shape on the teapot modeled using the marching-cubes method and the proposed method. The results indicate that some of the shapes were computed erroneously in the marching-cubes method. The meshes were also more irregular for the marching-cubes method due to the combined effect of the aforementioned noise and resolution problems. For all the above models obtained using the proposed method, the quality of the models is still unsatisfactory. The primary problems are that many narrow and sharp triangular faces exist on the models, and many faces are not connected smoothly. Therefore, a re-meshing and smoothing process should be implemented to improve the quality of the final model.

The 3D model generation using SFS is only the first step of the techniques developed for product presentation in e-commerce applications. As the mesh surface still contains lots of sharp edges, it needs a re-meshing process to improve the surface quality, which mainly includes the improvement of mesh regularity and surface smoothness. In addition, the 3D model should be textured using high quality object images to describe the outline appearance of an object in a more vivid manner. Two examples are presented below to demonstrate how the original triangular model is processed. Figure 20 shows several intermediate results of the proposed re-meshing algorithm for the case "cat doll". The proposed algorithm is based on a local re-meshing approach by Botsch et al. [1], in which the re-meshing process is divided into four steps: edge split, edge collapse, edge flip and vertex shift. In addition, the Laplacian mesh optimization [7] is also employed to improve the surface smoothness. Fig. 20(a) shows the original mesh; Fig. 20(b) shows the result after edge split is implemented; Fig. 20(c) shows the result after edge collapse is added; and Fig. 20(d) shows the result after edge flip is added. The final result in Fig. 20(d) clearly indicates that the mesh regularity and smoothness have been improved considerably. All long-and-narrow and tiny triangles in the original model have been erased. In texture mapping, an integrated algorithm based on the conformal mesh parametrization and a technique for direct texture mapping is developed. The conformal mesh parameterization [10, 12, 13] is employed to convert 3D meshes onto a 2D (UV) domain and keep the shape of the meshes on the UV domain. The direct texture mapping algorithm [4, 8] is primarily composed of three phases: grouping of 3D triangles, extraction of object image pixels, and placement of texture map pixels. Figure 21 shows the

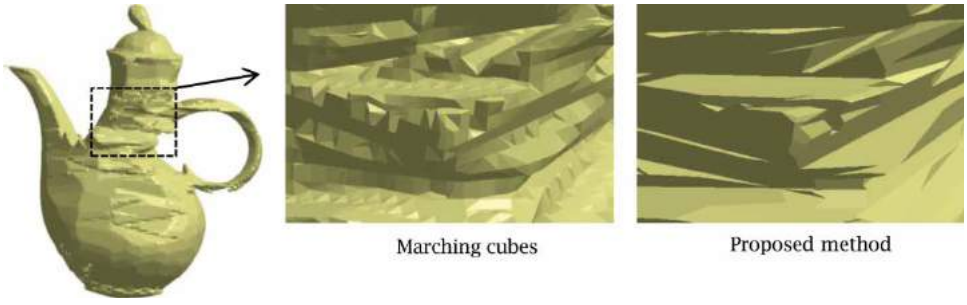

Figure 19. Combined noise and resolution problems for fine and composite shapes.

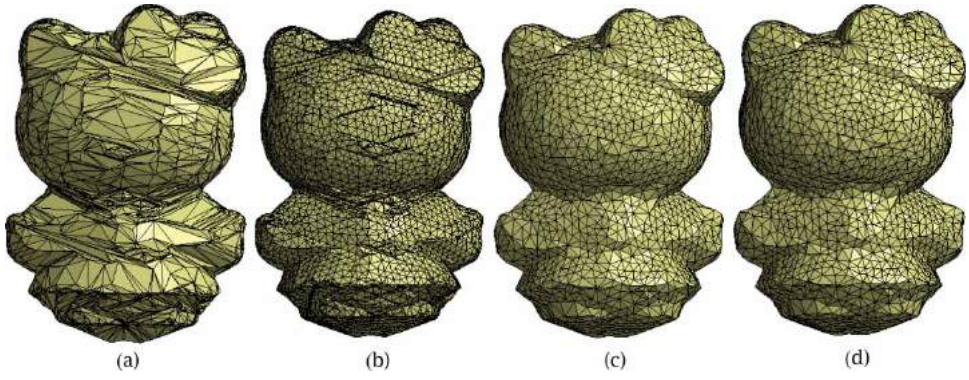

Figure 20. The proposed re-meshing algorithm for the case cat doll: (a) original mesh, (b) edge split, (c) edge split + edge collapse, (d) edge split + edge collapse + edge flip.

Figure 21. Texture mapping result in four different views for the cat doll model in Fig. 20(d).

result of texture mapping for the model in Fig. 20(d), in which four different views of the model with color texture are displayed. The current result shows the completeness of the entire process from multiple 2D images of the object to a complete 3D model with color texture. Several kinds of texture mapping algorithms are currently studied. A detailed discussion of the proposed re-meshing method and texture mapping method will be provided elsewhere.

5. Conclusions

In this study, we proposed a method for calculating 3D points and triangular meshes based on the shape-from-silhouette (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 and, hence, for obtaining 3D points. In addition, we developed an improved algorithm to reduce the overall computation time. A comprehensive study between the proposed method and the marching-cubes method was conducted. The proposed 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 concavities on an object because SFS considers only the bounding of a silhouette, which does not provide details about any concave areas on the object. Moreover, the surface quality of the triangular model is still not good enough for the purpose of rendering. A re-meshing strategy should be proposed to improve the mesh regularity as well the surface smoothness, without the deformation of the model as much as possible. In addition, a texture mapping technique could be developed to add texture to the model to make it visually pleasing, as is desired in e-commerce applications.

ORCID

Watchama Phothong http://orcid.org/0000-0002-3239-4564
Tsung-Chien Wu http://orcid.org/0000-0002-2299-7361
Jiing-Yih Lai http://orcid.org/0000-0002-0495-0826
Douglas W. Wang http://orcid.org/0000-0002-8039-5027
Chao-Yaug Liao http://orcid.org/0000-0001-8203-9520
Ju-Yi Lee http://orcid.org/0000-0002-2244-4863

References

[1] Botsch, M.; Kobbelt, L.; Pauly, M.; Alliez, P.; Levy, B.: Polygon Mesh Processing, 2010, 151–179. http://dx.doi.org/10.1201/b10688-10
[2] 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
[3] 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
[4] Genc, S.; Atalay, V.: Texture Extraction from Photographs and Rendering with Dynamic Texture Mapping, Proceedings 10th International Conference on Image Analysis and Processing, 1999, 1055–1058. http://dx.doi.org/10.1109/ICIAP.1999.797737
[5] 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
[6] Milne, P. S.: Visual Hulls for Volume Estimation: A Fast Marching Cubes Based Approach, M. S. Thesis, University of Cape Town, 2005
[7] Nealen, A.; Igarashi, T.; Sorkine, O.; Alexa, M.: Laplacian mesh optimization, Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, 2006, 381–389. http://dx.doi.org/10.1145/1174429.1174494
[8] 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

[9] 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

[10] Rhaleb, Z.; Lévy, B.; Seidel, H.: Linear angle based parameterization, Fifth Eurographics Symposium on Geometry Processing-SGP 2007, 2007, 135–141. http://dx.doi.org/10.2312/SGP/SGP07/135-141

[11] Sablatnig, R.; Tosovic S.; Kampel, M.: Next view planning for shape from silhouette, Proceedings of the 8th Computer Vision Winter Workshop, 2003, 77–82.

[12] Sheffer, A.; Sturler, E, de.: Parameterization of Faceted Surfaces for Meshing using Angle-Based Flattening, Engineering With Computers, 17(3), 2001, 326–337. http://dx.doi.org/10.1007/PL00013391

[13] Sheffer, A.; Lévy, B.; Mogilnitsky, M.; Bogomyakov, A.: ABF++: fast and robust angle based flattening, ACM Transactions on Graphics, 24(2), 2005, 311–330. http://dx.doi.org/10.1145/1061347.1061354

[14] 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

[15] 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