A Complete Visual Hull Representation Using Bounding Edges

Mohammad R. Raeesi N. and Q.M. Jonathan Wu

Electrical and Computer Engineering Department, University of Windsor,
401 Sunset Ave., ON, Canada
{raeesim,jwu}@uwindsor.ca

Abstract. In this article, a complete visual hull model is introduced. The proposed model is based on bounding edge representation which is one of the fastest visual hull models. However, the bounding edge model has fundamental drawbacks, which make it inapplicable in some environments. The proposed model produces a refined result which represents a complete triangular mesh surface of the visual hull. Further, comparison of the results by the state-of-the-art methods shows that the proposed model is faster than most of modern approaches, while the results are qualitatively as precise as theirs. Of interest is that proposed model can be computed in parallel distributively over the camera networks, while there is no bandwidth penalty for the network. Consequently, the execution time is decreased by the number of the camera nodes dramatically.

Keywords: Visual Hull, 3D Reconstruction, Shape From Silhouette (SFS), Bounding Edges.

1 Introduction

There are many applications such as security and surveillance which need to localize, recognize and reconstruct the objects as well as track them. Although many approaches including marker-based tracking have been proposed for these applications, some of them are not applicable in many environments, for example it is not possible to use the marker-based approaches for surveillance application in public places. The best applicable approach is using the vision networks because they are relatively cheaper and can be configured easily [1].

The volumetric description from multiple views is first described by Martin and Aggarwal [2]. In vision networks, applications recover the 3D shape of the objects based on the captured images from different calibrated views of the object. Most of the existing algorithms use the silhouette concept to get the information from the images. Silhouette is a binary image in which the pixels are labeled either foreground or background. Baumgart [3] first considered silhouettes to approximate a polyhedron representation of the objects. Silhouette images are very efficient for vision networks in case of communication, because their size is much smaller than the size of the raw images. For example, a 2000x1500 color image is approximately 400KB, while a same size silhouette is less than 8KB, without any compression.

G. Qiu et al. (Eds.): PCM 2010, Part I, LNCS 6297, pp. 171–182, 2010.

The constructed objects of the silhouette is called visual hull. Visual hull concept has been first defined by Laurentini [4]. Based on the silhouettes information, visual hull is the best approximation of the interesting object. Because visual hull is constructed from the silhouette images, it is also called Shape from Silhouette (SFS). Visual hull is the maximal one of the objects which has the same silhouettes as the given ones. By increasing the number of different views, the constructed visual hull will be tighter. The greater the number of the views, the more precise the approximated visual hull. The visual hull applications and the resulted models are very sensitive to silhouette noise and camera calibration errors.

The goal of all the algorithms in this field is to construct a visual hull H from the input set of silhouette images from different points of view $\{S_k | k = 1, \dots, K\}$, where K is the number of cameras in the network.

Camera calibration is an important issue in vision network which is out of the scope of this article. There are many works done to calibrate the cameras. It is considered that the cameras are calibrated, and there is a function $\Pi_k(P)$: $\mathbb{R}^3 \longrightarrow \mathbb{Z}^2$, which maps a 3D space point P to a 2D pixel coordinate p in the k^{th} image plane.

There are many visual hull models which are reviewed in section 2. Afterwards, the proposed visual hull model is described, followed by its steps in section 3. In section 4, the experiments are evaluated and compared with the other models. Finally, the last sections are conclusion and future works.

2 Existing Visual Hull Representations

In existing approaches for modeling the objects, two categories are popular; voxel-based approaches and surface-based (polyhedron) ones. The first one models the objects by a collection of elementary cells. These cells are called voxels (volumetric cells), which are first suggested by Martin et al. [2]. The discrete volumetric representation generates some quantization and aliasing artifacts on the resulting model. The voxel-based approach has been improved by introducing octrees, which have been first considered by Jackins et al. [5] as an efficient geometric representation. Octrees are tree-structured representations which are used to model the volumetric data. The precision of the octree model is better than the voxel one, if their storage spaces are that same. The time needed to construct a voxel model is greater than what the octree model needs, because it evaluates more geometrical cells.

The second category of the popular modeling approaches is surface-based one. In this category, a polyhedron model of the object is produced by intersecting the silhouette cones. The surfaces of the polyhedron are the visual cone patches, its edges are the intersection curve between two silhouette cone, and the vertices are the points where more than two silhouette cones intersect.

Lazebnik et al. [6] proposed two representations for the visual hull, the rim mesh and the visual hull one. They defined the rim mesh by its vertices (the frontier points), edges (the segment between successive frontier points), and the faces (the surfaces bounded by the edges). Because the rim mesh depends only on the ordering of the frontier points, the rim mesh is topologically more stable, while the visual hull meshes reliably recover the geometry information. Buehler et al. [7] proposed the real time representation of the polyhedral visual hulls. It is suited to be computed by the graphics hardware. The most important issue is that they assumed each silhouette is a 2D

polygon. Franco et al. [8] proposed a fast algorithm to represent the best polyhedral visual hull. They first computed a coarse approximation of the visual hull by retrieving the viewing edges. Then, the surfaces of the mesh are generated. Finally, the faces of the polyhedron are identified.

There are other approaches to model the visual hull. One of the compact representations of the visual hull is image-based visual hull. The visual hull is represented by the rays through the view points of the image plane. Buehler et al. [9] defined the image-based representation as a two dimensional, occupancy intervals samples. The samples are the intervals of the rays which are inside the visual hull. In other word, the intervals of the ray which intersect all the other silhouette images are stored in the samples. So for each pixel, the list of its corresponding intervals is stored. The list for a pixel is empty, if it is a background pixel.

The image-based representation has many advantages in comparison to other models. Its storage and computational requirements are much less because its algorithm is a simple geometrical computation. Moreover it makes the rendering simple. Since it has two discrete dimensions and one continuous dimension, its resolution is higher than the resolution of the voxel-based representation, but it is an incomplete visual hull representation [9]. A visual hull is complete, if it has all the geometrical information of its shape. Because there is no information for the parts between the two successive occupancy intervals, the image-based model is called an incomplete model. Matusik et al. [10] proposed an image-based approach to represent the visual hull. They defined the visual hull approximation as carving away the regions of space where the object is not.

The next visual hull representation is bounding edges model. This representation has been first introduced by Cheung [11]. He defined the bounding edge representation as parts of the rays from the viewing point through the contour of the corresponding silhouette image which intersect all the other silhouette images. Bounding edges are very similar to the image-based representation. Like image-based model, it calculates the intervals of the rays from the view point through the silhouette pixels. But in contrast to image-based representation, it considers only the silhouette contour pixels instead of all the silhouette pixels. Bounding edges lie exactly on the surface of the visual hull, but they are incomplete [12].

Based on the existing visual hull models and their strengths and drawbacks, the new model is proposed which is described in the next section.

3 Proposed Model

All the visual hull models have some weaknesses. The volumetric models are not applicable in some application because of the quantization errors. The surface-based models suffer from the complexity of the computation it needs as well as the run time. The bounding edge and image-based models are incomplete. Moreover, the image-based model is view dependent. Fortunately, it is possible to overcome disadvantages by applying other algorithms to improve the final results. We found that it is possible to produce a complete visual hull model based on the bounding edge visual hull. This section describes the ideas and algorithms which are used in the new model.

The base contribution for the proposed model is to provide a complete visual hull representation based on the incomplete representation fundamentals. The bounding edge representation is an incomplete representation, but it is not view dependent because it is applied on all points of view. Based on the bounding edge model, we can provide an incomplete, but accurate visual hull representation of the 3D object. As mentioned before, the bounding edge model is efficient in execution time as well as storage space requirement. Our contribution is to provide a surface mesh over the incomplete visual hull model, which results a complete and accurate 3D triangular mesh representation of the object in an acceptable time instance.

Our proposed visual hull algorithm consists of the following four steps:

1. Applying a modified bounding edge model on the set of the silhouettes.
2. Provide bounding surfaces based on bounding edges for each viewpoint.
3. Merge the bounding surfaces to produce the final visual hull mesh.
4. Applying a re-meshing algorithm to improve the quality of the final mesh.

All the mentioned steps are described in the following subsections.

The idea for this work is motivated from Projective Visual Hulls which is published by Lazebnik et al. [13]. They considered the cone strips of the surface of the cones as the boundaries of the visual hull. They provide a mesh based on the edges and points they recover from the visual cones. The edges are intersection curves between two visual cones, and the points are frontier points and intersection ones. As the first step of their work, they provide the surface of the cone strips from each point of view. Then the cone strips provide the final visual hull as a triangular mesh. Their work is based on oriented projective differential geometry, which transfers the data from the 3D space to 2D one.

The idea taken from the projective visual hull model is to provide a final visual hull mesh based on the bounding surfaces. The bounding surfaces are the surfaces produced based on the information from bounding edge model. In overall view, our model is similar to Projective Visual Hull. The outputs of the steps are similar to each other, but not the same. The output of the first steps for both model are the geometrical information recovered from silhouette images. In our model, the information are bounding edges, while in Projective model, it is the intersection curves and points. More important, the details of each step are completely different. For example, the merging step merges the surfaces provided from each point of view for both models, but in different way, because their input information are not the same. However, the last step, refining the final model, is the same for both models.

3.1 Modified Bounding Edge Model

The first step of the algorithm is calculating bounding edges. The bounding edge model which is used in the proposed algorithm is different from the main bounding edge model in only one part. The difference is the information they record for each contour pixel. The method used to calculate the occupancy intervals are the same as what Matusik et al. [10] used for their image-based model.

The main bounding edge model works as follows. Each contour pixel p_i^k of the silhouette S_k is back projected to a 3D ray R_i^k which starts from camera center C_k and goes through the 3D position of the mentioned pixel coordinate p_i^k. The 3D ray

R_i^k is the position of all the 3D points P which are mapped to the corresponding contour pixel p_i^k of the silhouette S_k by function $\Pi_k(P)$,

$$R_i^k = \{P | \Pi_k(P) = p_i^k\} \quad . \tag{1}$$

The algorithm starts with a contour pixel and continues to its neighbor recursively, until algorithm reaches the start point. The index i for the contour pixels is based on the mentioned order, which can be clockwise or counterclockwise. In our experiment, we consider the counterclockwise order, in which the map of the object is always at the left hand side of the direction of traversing the contour points. In the next step, the 3D rays are projected to the all other silhouette planes, and intersected with the silhouettes. Finally, the intersection parts of the rays with all other silhouettes are returned to the 3D space, which are the occupancy intervals.

It is not necessary for the occupancy intervals to be complete. The occupancy intervals can consist of more than one segment, if there is at least one non-convex silhouette image. The intervals are saved for each contour pixel, as a set of segments. Each segment is considered as a pair of its endpoints, start and finish points. For each endpoint, only the distance to the corresponding camera center is saved which is 1D value (real number). Bounding edge E_i^k is shown by

$$E_i^k = \left\{ \left(SP_{i,m}^k, FP_{i,m}^k \right) \middle| m = \{1, \dots, M\} \right\} \tag{2}$$

where M is the number of segments of the bounding edge. $SP_{i,m}^k$ and $FP_{i,m}^k$ are the distance from the start point and finish point of the m^{th} segments to the camera center C_k, correspondingly.

The difference between our proposed model and the main bounding edge model is the information recorded for each occupancy interval. The main model records only 1D value(real number) for each endpoint of each occupancy interval. In our model more information is recorded for each endpoint of occupancy intervals. It includes the 1D value, the silhouette which intersects the occupancy interval at the corresponding endpoint and the pixel of the silhouette which cuts the occupancy interval at the position of endpoint. Consider an occupancy interval$\left(SP_{i,m}^k, FP_{i,m}^k \right)$. When a 3D ray R_i^k is projected to a silhouette plane $S_{k'}$, the endpoints of the intersection parts of the projected ray with the silhouette $S_{k'}$ are its contour pixels. The $SP_{i,m}^k$ and $FP_{i,m}^k$ are back-projection of the contour pixels to the 3D ray R_i^k. In our model, we record references to the silhouette $S_{k'}$ and to its corresponding contour pixels.

This modification does not affect the run time of the main model, because it is similar to the main bounding edge model and only keeps more information. So it needs more storage space than the main model. For each endpoint in the main model, there is only 1D value, but in the modified model, each endpoint needs to have sufficient space for 1D value, the silhouette reference, and the pixel position.

Fig. 1 shows the resulted bounding edge model and the corresponding contour map for the first view of the Dinosaur dataset. The contour map is a diagram for which the x-axis is the contour pixels in their order and the y-axis is the occupancy intervals in term of their distance to the camera center.

There are two types of discontinuities in contour map. The first one is the discontinuity for inconsistent contour pixels. Cheung et al. [14] defined a consistency concept for the set of silhouette images. The set of silhouette images is consistent, if there is at

least one non-empty object O that exactly explains all the silhouette images $\{S_k\}$ which means that the projections of the volume to the silhouette planes fit the silhouettes, that is

$$\exists O \; \forall k \in \{1, \ldots, K\} \quad \Pi_k(O) = S_k \; . \tag{3}$$

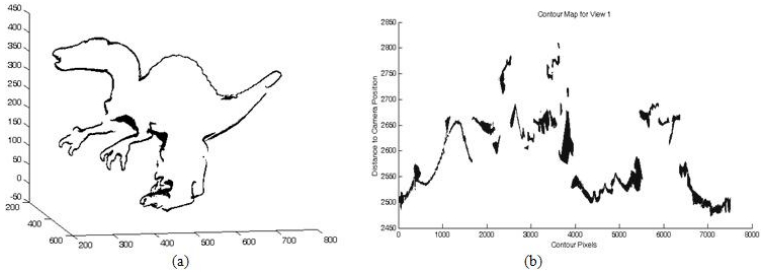

Fig. 1. (a) The bounding edge model for the 1st model and (b) corresponding contour map

The inconsistent pixels are those pixels whose back-projected 3D ray has no intersection with all the other silhouettes. This type of discontinuity is removed for the final visual hull automatically, because the rays from different points of view cover the discontinuity. However, the inconsistent pixels can be removed as a preprocess step for the model. The preprocess step first finds the inconsistent pixels and removes them from the silhouettes. Like bounding edge step, the preprocess algorithm starts from a contour pixel, and traverses the contour pixels in a way that the silhouette is located on the left hand side. In processing each pixel, it checks whether the corresponding 3D ray has intersection with all the other silhouettes. If there is any intersection, it goes for the successor contour pixel. Otherwise, it removes the current pixels from the silhouette and then finds a new successor for the preceding pixel. This routine is continued until the starting point is reached.

The Table 1 shows the result of applying preprocess algorithm on the first 8 silhouettes from the Dinosaur dataset. The result shows that the percentage of inconsistent pixels is less than 0.5% for each point of view. After the preprocess step, the proposed algorithm will apply to the consistent silhouette set.

Fig. 2 shows the input silhouette and the differences between the input silhouette for the first view and the consistent one resulted by applying preprocess. The difference image contains the inconsistent pixels which are 2444 for the first point of view. It has the greatest number of inconsistent pixels because of the relative position of the object to the corresponding camera center.

Table 1. Numbers of inconsistent points for the first 8 views of the Dinosaur dataset

View	1	2	3	4	5	6	7	8
Silhouette Points No.	604,566	429,018	378,636	588,082	627,430	480,970	394,818	622,285
Inconsistent Points No.	2,444	949	608	977	1,150	315	917	1,310
Percentage (%)	0.40	0.22	0.16	0.16	0.18	0.06	0.23	0.21

The second type of discontinuity is duo to the self-occlusion. Since the interesting object here, dinosaur toy, is a self-occluded object, some parts of its body are occluded in some point of view. The occlusion causes some discontinuities in the bounding edge model. As it can be seen clearly in the 3D representation of the resulted bounding edge model, Fig. 1a, the arms of the dinosaur, for example, are not connected to its body, and also there is no information for the part of its stomach which is occluded by arms. These discontinuities can be seen in the contour map as well. Since the occluded parts of this view are visible from other points of view, discontinuities are recovered for the final visual hull by the occupancy intervals from the other viewpoints. It should be mentioned that the occluded parts of the 3D object which are not visible in all views do not make any discontinuity in contour map.

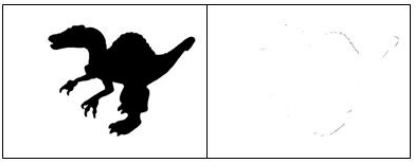

Fig. 2. The silhouette of the 1st view of Dinosaur dataset (left) and the inconsistent pixels which are the difference between the input silhouette and the consistent one (right)

3.2 Bounding Surfaces

After computing the bounding edge information, it is time to produce the mesh over the computed bounding edges. This job is done for each point of view individually. A surface is generated using a triangular mesh algorithm. The input for this step is a contour map, and the output is a 3D triangular mesh surface. The algorithm considers the gap between occupancy intervals of two successive contour pixels as the surface of the visual hull, if they have any intersection with each other. If a gap between two occupancy intervals are considered as a part of surface, then two triangles will be generated which have one occupancy interval as a side and one endpoint from other occupancy interval as a vertex. Consider two successive contour pixels p_i^k and p_{i+1}^k. For each segment of their occupancy intervals, the endpoints are evaluated. Consider the m^{th} segment of the occupancy interval for point p_i^k and the n^{th} segment of the occupancy interval for the next pixel. If one of the endpoints of each of them is located between the endpoints of the other one, the gap between these two segments is considered as a part of the strip mesh surface. For instance, if $SP_{i,m}^k$ which is a real number is greater than $SP_{i+1,n}^k$ and smaller than $FP_{i+1,n}^k$, then two triangles are added to the strip mesh surface. These triangles are triple points $\left(SP_{i,m}^k, FP_{i,m}^k, FP_{i+1,n}^k\right)$ and$\left(SP_{i,m}^k, FP_{i+1,n}^k, SP_{i+1,n}^k\right)$. To have the best triangular mesh, based on the positions of the endpoints, the new points may be added. To select the occupancy intervals for providing the surfaces, only the 1D value of the endpoints are used. The other information will be used for the next section to merge the surfaces.

3.3 Merging Bounding Surfaces

The next step is merging the resulted bounding surfaces. To merge the surfaces, the extra information recorded in the first step is used. We call both the start point $SP_{i,m}^{k}$ and finish point $FP_{i,m}^{k}$ as the endpoints $EP_{i,m}^{k}$. As mentioned before, an endpoint $EP_{i,m}^{k}$ of any segments of any occupancy interval has a reference to the silhouette $S_{k\prime}$ which has an intersection with one of its contour pixels $p_i^{k\prime}$. Because $p_i^{k\prime}$is a contour pixel of silhouette $S_{k\prime}$, it should have an occupancy interval $E_i^{k\prime}$ for the bounding edges of the silhouette $S_{k\prime}$. This interval crosses the endpoint$EP_{i,m}^{k}$. Endpoint $EP_{i,m}^{k}$ can be positioned on an endpoint of a segment of $E_i^{k\prime}$or on the middle of a segment.

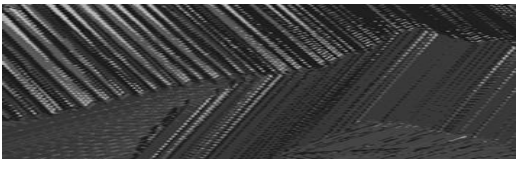

Fig. 3. Intersection of the occupancy intervals form different viewpoints

Fig. 3 shows a part of the final triangular mesh, in which some endpoints are the endpoints for another point of view (right hand side of the figure) and others are the middle points (left hand side of the figure). Based on the concept mentioned above, it can be concluded that each endpoint of occupancy intervals at least exists in one bounding edge model from different point of view. So by finding these points, it is possible to merge the surfaces. By this algorithm, the number of the points of the merged surface is much less than the points of the overall strip surfaces. The experiments show that the number of the points is decreased by 30 to 40 percent. In the first consideration, it seems it should be decreased by more than 50 percent, but it is not. Some endpoints are located on the middle of another occupancy interval. Because middle points are not counted as endpoints, the decreasing amount of the point number is less than 50 percent. The decreasing percentage of the point number depends on the 3D object and the relative positions of the cameras.

3.4 Re-meshing

The final step of the proposed model is refining the resulted mesh. Because of the lack of the vertices along the occupancy intervals, which are used to produce the triangular bounding surface mesh, the triangles are thin and long. To refine the triangles, a set of edge split operations (for long edges), edge collapse operations (for short edges) and edge swap operations (to guarantee that each vertex has a degree close to six) are applied on the final mesh. After applying the re-meshing step, we will have a refined complete triangular mesh of the visual hull.

4 Experiments

To show the quality of the proposed model, it is applied to some datasets described here. The results have been shown in the next subsection followed by an evaluation

part. To show the effectiveness of the algorithm, complex 3D datasets have been selected. These datasets are the 3D Photography datasets [15] and Middlebury Multi-View datasets [16]. Each dataset of 3D Photography collection has 24 images from 24 points of view, which are calibrated using Intel's OpenCV package [17]. Moreover, the contour information has been provided as unconnected 2D pixels. From Middlebury dataset, we select DinoSparseRing dataset to evaluate the performance of our model. DinoSparseRing dataset has 16 images from different viewpoints as well as suggestion steps to produce silhouette information.

There is one step before applying the proposed model which is producing the consistent silhouette set for each dataset based on provided information.

4.1 Results

Fig. 4 shows the bounding mesh surfaces resulted from the first 8 views of Dinosaur, Predator and DinoSparseRing datasets. Each image shows the bounding surface from one viewpoint. As it can be seen clearly, the strip surfaces are not connected and there are some discontinuities in them.

Fig. 4. Bounding surfaces resulted for the first 8 views of Dinosaur (1st row), Predator (2nd row), and DinoSparseRing (3rd row) datasets

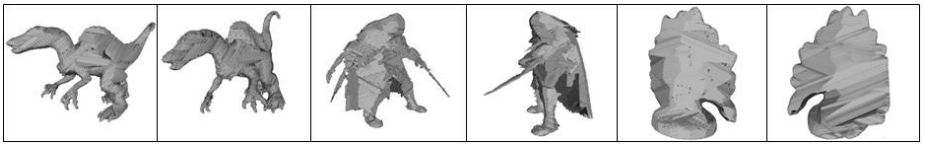

Fig. 5. Final triangular meshes for Dinosaur, Predator and DinoSparseRing datasets

Fig. 5 shows the merged surface of the bounding surfaces. The surfaces are connected and the discontinuities have been removed from the mesh. Number of vertices in the overall surface and merged surface before re-meshing for each dataset has been shown in Table 2. By merging the surfaces, number of vertices is decreased significantly. For example, for Dinosaur dataset, it has been decreased by 40%. It is true that some vertices in the final mesh are removed because they are identical in two or more points of view.

Table 2. Number of vertices in the all surfaces versus the merged surfaces before re-meshing

Dataset	All Surface	Merged Surface	Percentage (%)
Dinosaur	432,422	261,017	60.36
Predator	388,406	258,726	66.61
DinoSparseRing	126,772	80,063	63.16

As mentioned before, increasing number of views improves the accuracy of the final result. So determining the number of views required to obtain an acceptable result depends only on the applications and their definition of being acceptable. Fig. 6 shows the results for different number of views for DinoSparseRing dataset. As it can be seen clearly, the final results for greater number of views are more precise.

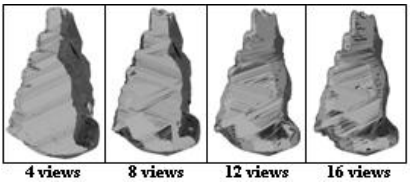

Fig. 6. Final results for different number of views for DinoSparseRing dataset

4.2 Comparison and Evaluation

Since the proposed model is complete and has a triangular mesh surface, to compare and evaluate the results, complete triangular models should be considered. For this article, the projective visual hull model and the last two versions of Exact Polyhedral Visual Hulls [8] are selected for comparison. The results for other models are taken from Lazebnik et al. [13] which are produced by running the algorithms on an Intel Pentium IV desktop with a 3.4GHz processor and 3GB of RAM. To have a consistent comparison, the proposed model is executed on the same machine.

The results have been shown in Table 3. It should be mentioned here that the images are the results of model of first 8 views of the datasets, while the times mentioned in Table 3 are the execution time of the model over all views of the datasets to make the comparison possible. As it can be seen clearly, the proposed model is faster than the Projective Visual Hull and the first version of EPVH, while it is not as fast as EPVH 1.1.

Table 3. Execution time of the final visual hull model produced by different models in second

Dataset	EPVH 1.0	EPVH 1.1	Projective	Proposed
Dinosaur	6,329.5	138.0	513.4	479.3
Predator	5,078.2	136.0	737.2	647.9

Since there is not any ground truth for the ideal visual hull model, it is not possible to compare the results quantitatively, but it can be said that the results of the proposed model are qualitatively as accurate as the mentioned existing algorithms. This is evaluated by checking the critical parts of the interesting objects which are so complex. One of these critical parts is the connection of the dinosaur's hand to its body. It should be mentioned

again that the figures are resulted based on only the first 8 views of each datasets, while other algorithms used all views.

Comparing the required time, the proposed model is similar to the Projective Visual Hull model. The main step of Projective model which takes much amount of time is calculating the first generation of information, producing the 1-skeleton of the 3D object. For Dinosaur dataset, for instance, producing 1-skeleton takes 318.9 seconds, while the time needed for the triangulation step is 76.8 [13]. The proposed model works the same as Projective Visual Hull representation. The execution time for producing the 3D mesh surfaces and merging them takes only 6.8 seconds for Dinosaur dataset, which is much less than 472.5 seconds for the first step. Another issue is that our merging step is much faster than the merging step for Projective model.

The most important advantage of our model is that it can be computed in distributed manner. If the camera nodes have processor units, they can participate in the first step of the algorithm. Because the first step is based on each viewpoint independent to other views, it can be done by each camera node. So the execution time for producing the bounding surfaces will be divided to the number of camera nodes. In this case, the overall execution time will be decreased dramatically. For instance, the final result of Dinosaur dataset will be obtained in less than 30 seconds. The merging step can be executed by the main server for centralized camera networks or by any of the camera nodes in the network or by all of them simultaneously, which depends on the application. The communication over the network is not an issue because the input for the first step is silhouette images and the output is the occupancy intervals for contour pixels of the silhouettes which are so efficient for network communication. The results of the distributed programming are compared with the sequential programming in Table 4. The large difference between the execution time of DinoSparseRing dataset and others is because of the number of images for each dataset and the size of the images shown in Table 4.

Table 4. Execution time sequentially versus distributedly in second

Datasets	Dinosaur 24 views - 2000×1500		Predator 24 views - 1800×1800		DinoSparseRing 16 views - 640×480	
	Sequential	Distributed	Sequential	Distributed	Sequential	Distributed
Bounding Surfaces	479.3	21.97	647.9	28.58	37.92	2.87
Merging Surfaces	6.8	6.8	7.6	7.6	2.8	2.8
Overall	486.1	28.77	655.5	36.18	40.72	5.67

5 Conclusion

In this paper, a new simple yet versatile model for visual hull representation is proposed. It is based on bounding edge model which is one of the fastest models. The execution time of proposed model is close to the time required for bounding edge model. However the storage requirement is more than what needed for the bounding edge model, the final result is compact relatively. It only keeps vertices and faces information of the triangular mesh.

In comparison to the state-of-the-art algorithms, the execution time and storage space is satisfactory. In most cases, our model is faster. Moreover, the final result is

qualitatively as accurate as modern approaches. The main advantage of our model is that its computation can be divided to the camera nodes over the camera network, while it does not need high communication bandwidth. By computing this job in parallel, the execution time is decreased dramatically.

References

1. Cheung, K.M., Baker, S., Kanade, T.: Shape-From-Silhouette Across Time Part II: Applications to Human Modeling and Markerless Motion Tracking. International Journal of Computer Vision 63(3), 225–245 (2005)
2. Martin, W.N., Aggarwal., J.K.: Volumetric Description of Objects from Multiple Views. IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI) 5(2), 150–158 (1983)
3. Baumgart, B.G.: A Polyhedron Representation for Computer Vision. In: AFIPS National Computer Conference (1975)
4. Laurentini, A.: The Visual Hull: A New Tool for Contour-based Image Understanding. In: 7th Scandinavian Conference on Image Analysis, pp. 993–1002 (1991)
5. Jackins, C.L., Tanimoto, S.L.: Oct-trees and Their Use in Representing Three-dimensional Objects. Computer Graphics and Image Processing 14, 249–270 (1980)
6. Lazebnik, S., Boyer, E., Ponce, J.: On Computing Exact Visual Hulls of Solids Bounded by Smooth Surfaces. In: CVPR 2001 (December 2001)
7. Buehler, C., Matusik, W., McMillan, L.: Polyhedral Visual Hulls for Real-time Rendering. In: Eurographics Workshop on Rendering (2001)
8. Franco, J.-S., Boyer, E.: Exact Polyhedral Visual Hulls. In: Fourteenth British Machine Vision Conference (BMVC), Norwich, UK, pp. 329–338 (September 2003)
9. Buehler, C., Matusik, W., McMillan, L., Gortler, S.: Creating and Rendering Image-based Visual Hulls. Technical Report. MIT-LCS-TR-780. MIT (1999)
10. Matusik, W., Buehler, C., Raskar, R., Gortler, S.J., McMillan, L.: Image-based Visual Hulls. In: SIGGRAPH 2000 (July 2000)
11. Cheung, G.: Visual Hull Construction, Alignment and Refinement for Human Kinematic Modeling, Motion Tracking and Rendering. Doctoral dissertation, Technical Report CMU-RI-TR-03-44, Robotics Institute, Carnegie Mellon University (October 2003)
12. Cheung, G., Baker, S., Kanade, T.: Visual Hull Alignment and Refinement Across Time: a 3D Reconstruction Algorithm Combining Shape-Frame-Silhouette with Stereo. In: CVPR 2003, Madison, MI (2003)
13. Lazebnik, S., Furukawa, Y., Ponce, J.: Projective Visual Hulls. International Journal of Computer Vision 74(2), 137–165 (2007)
14. Cheung, K., Baker, S., Kanade, T.: Shape-From-Silhouette Across Time Part I: Theory and Algorithms. International Journal on Computer Vision 62(3), 221–247 (2005)
15. 3D Photography Dataset. Beckman Institute and Department of Computer Science, University of Illinois at Urbana-Champaign, `http://www-cvr.ai.uiuc.edu/ponce_grp/data/mview/`
16. Middlebury Multi-View Datasets. Middlebury College, Microsoft Research, and the National Science Foundation, `http://vision.middlebury.edu/mview/`
17. Intel's OpenCV library written in C programming language, `http://sourceforge.net/projects/opencvlibrary/`