A Robust and Accurate Method for Visual Hull Computation

Peng Song, Xiaojun Wu and Michael Yu Wang

***Abstract*— A novel method for computing visual hull mesh from a sequence of silhouettes is addressed in this paper. Firstly we use genetic algorithm to estimate the bounding box of the scene object from the silhouettes. Secondly, we develop a method for computing the octree of visual hull through a new projection test strategy to determine whether a voxel locating outside, on or inside the visual hull. The projection test depends on a fact that the projection of a voxel in an image is the same as the convex hull of the projections of the cube's 8 vertices. Finally, the visual hull can be extracted from the Octree. In order to get smooth visual hull mesh, we compute the 2D distance to the silhouette, and use it to evaluate the 3D distance to the real visual hull surface. Experiments with several real data sets are presented to validate our algorithm.**

I. INTRODUCTION

The visual hull [1] is an outer approximation of the observed object and can be constructed by intersection of the visual cones formed by back-projecting the silhouettes in the corresponding images, shown in Fig. 1. Defined in full generality, the visual hull is the maximal shape consistent with an object's silhouettes as seen from any viewpoint in a given region. The visual hull is not guaranteed to be the same as the original object since concave surface regions can never be distinguished using silhouette information alone. However, the visual hull mesh reconstructed from silhouettes still has many applications in the field of stereo vision, markless motion capture, shape analysis etc. It offers a rather complete description of the scene solid, especially for smooth curved objects. In many cases, the visual hull mesh can be directly fed to some 3D applications as a showcase, or used as a good initial estimation for various surface reconstruction algorithms [2],[3].

Computing the visual hull from a sequence of images is a very well known problem in computer vision and computer graphics. Different approaches exist, depending on the type of output, way of representation and fidelity to the theoretical visual hull. However, there are two major approaches:

- Polyhedral approach: It is calculating a polygonal surface by intersecting silhouette cones, computed easily by back-projection polygonal approximations of silhouette contours of the object, which was first realized in [4]. Since direct 3D intersections suffer from numerical instabilities, recent studies have improved robustness by calculating the intersections on 2D image planes and back-projecting the results [5]. More recently, researchers have presented efficient algorithms that avoid general 3D intersections by taking advantage of epipolar geometry [6],[7].
- Volumetric approach: It represents an alternative approach to visual hull construction. In this case, the 3D space is divided into elementary cubic elements (i.e., voxels) and projection tests are performed to label each voxel as being inside, outside or on the boundary of the visual hull. This is done by checking the contents of its projections on all the available binary silhouette images. The output of volumetric methods is either an octree [8],[9], whose leaf nodes cover the entire space or a regular 3D voxel grid [10],[11]. Coupled with the marching cubes algorithm [12], a surface can be extracted. A drawback of the approaches is that the octree offers only a binary description of the visual hull, rather than a continuous scalar field. Therefore, it is insufficient for the marching cubes algorithm to smoothly interpolate vertex positions during the extraction of surface mesh triangles. To solve this problem, we replaced the binary value at each vertex of the voxels with an estimated value that represents the 3D distance to the visual hull surface [13].

The most important advantage of volumetric-based approaches is their robustness. This paper proposes a new volumetric method to compute visual hull meshes, which is also based on the polygonization of octree construction by using the marching cubes algorithm, but uses a different projection test method to check whether a voxel locates outside, on or inside the visual hull, which can offer a robust scheme to create a visual hull. We begin in Section 2 with background. In Section 3 we describe our approach to compute visual hull. In Section 4 we show some results of our algorithm operating on a sequence of images. We close in Section 5 with a conclusion.

II. BACKGROUND

As mentioned above, our method is based on the polygonization of octree construction by using the marching cubes algorithm. In this section we review the Octree/Marching cubes algorithm and introduce some projection test methods.

A. Octree and marching cubes algorithm

An octree is a tree data structure in which each internal node has up to eight children. Each leaf-node of an octree corresponds to an actual volume element, also termed as

This project is partially supported by Natural Science Foundation of China (NSFC No.50805031)

P. Song is with postgraduate of Shenzhen Graduate School, Harbin Institute of Technology, 518055, China songpenghit@163.com

X.J. Wu is with faculty of the Division of Control and Mechatronics Engineering, Shenzhen Graduate School, Harbin Institute of Technology, 518055, China, wuxj@hitsz.edu.cn

M.Y. Wang is with the Mechanical and Automation Engineering Department, Chinese University of Hong Kong, Shatin, N.T., Hong Kong, China, yuwang@mae.cuhk.edu.cn

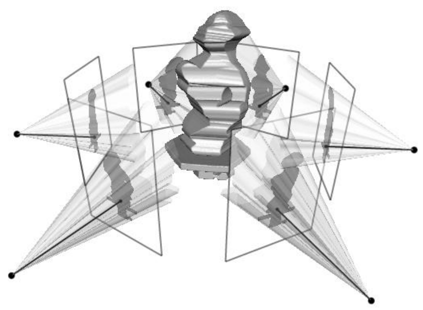

Fig. 1: The intersection of silhouette cones defines an approximate geometric representation of an object called the visual hull [3].

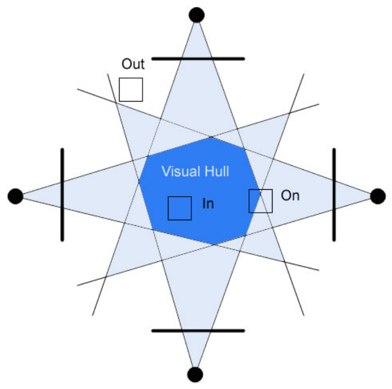

Fig. 2: Three types of voxels in octree based visual hull reconstruction: on, in, and out of the visual hull.

voxel, in 3D space. The root node is thus the bounding volume of the object to be reconstructed. Octrees are most often used to partition a three dimensional space by recursively subdividing it into eight octants.

An octree of an object can be reconstructed from the silhouettes through recursive subdivision and projection tests. The process usually begins with a single voxel, which is the 3D bounding box of the object. Each voxel is projected onto all the images and tested against the silhouettes. The test result classifies the voxel as being inside, outside or on the boundary of the visual hull (see Fig. 2). Among the three types of voxels, only voxels on the boundary of the visual hull contain the potential visual hull boundary and are subject to further subdivision until the maximum allowed number of subdivision is reached.

Once the octree construction of the visual hull is computed, marching cubes algorithm is used to extract the visual hull mesh from the octree.

B. *Some projection test methods*

When building octree construction of visual hull, the most important part is projection test, which is a process to check the projection of a voxel on all the available binary silhouette images. The test result classifies the voxel as being inside, outside or on the boundary of the visual hull. Specifically, if the projection of the voxel is in all the silhouettes, the corresponding voxel is inside the visual hull surface; if the projection is completely out of at least one silhouette, its type is out; else, the voxel is on the visual hull surface. In practice, the computation of the exact projection of a voxel in an image is complicated and error prone. Some easy projection test methods have been proposed to determine the location of a voxel.

- R. Szeliski's projection test method [8]: A cube projected into the image plane will in general form a six-sided polygon. Szeliski used a coarser test based on the hexagon's bounding box, which may sometimes fail to detect a true inclusion or exclusion. Since the goal of Szeliski's algorithm is to obtain a quick and rough model of an object in close to real time, this method is adaptable.
- C. H. Esteban's projection test method [3]: Esteban solves this problem by oversampling the edges of the voxel up to the maximum octree level such that the voxel type decision is based on the state (in or out) of the intermediate sampled points along the edges and the corners. That is: (1) If all these points of a voxel are out of the visual hull, the voxel's type is out. (2) If all these points of a voxel are in the visual hull, the voxel's type is in. (3) If parts of these points of a voxel are in the visual hull, the voxel's type is on. In fact, his method dos not solve the problem completely. Since when a part of the object only intersects a face of the cube, this method is unavailable.
- M. Potmesil's projection test method [9]: Potmesil's projection test method is: compute the exact projection of a voxel and do intersection test with the silhouettes, which is similar to our method. However, we use a more efficient method to compute the exact projection of a voxel. This will be introduced in section 3.2.

In this paper, we propose a new projection test scheme to overcome the aforementioned problem to construct a visual hull.

III. VISUAL HULL COMPUTATION

We compute the visual hull mesh in three steps: first, compute the 3D bounding box of an object. Then, build the octree construction of the visual hull, taking the 3D bounding box as a root node. At last, marching cubes algorithm is used to extract the visual hull mesh from the octree.

A. *Calculate 3D bounding box*

To build octree construction of visual hull, the 3D bounding box is needed as a root node. In practice, an accurate 3D Bounding Box can improve the precision of the result mesh. We calculate the 3D bounding box only from a set of silhouettes and the projection matrices. This can be done by considering the 2D bounding boxes of each silhouette. The bounding box of the object can be computed by an optimization method for each of the 6 variables defining the bounding box, which are the maximum and minimum of x, y, z.

We use 4 numbers to describe 2D bounding box of each image as in Fig. 3. Suppose there are N images to calculate

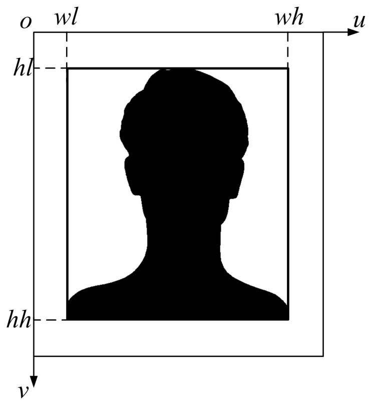

Fig. 3: 2D bounding box of a silhouette.

the 3D bounding box, P_i is the projection matrix of image i, and $wl_i, wh_i, hl_i, hh_i, (i = 1, 2, \cdots, N)$ define its 2D bounding box. The perspective projection equation is:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \frac{1}{K} * \begin{pmatrix} P_{11}^i & P_{12}^i & P_{13}^i & P_{14}^i \\ P_{21}^i & P_{22}^i & P_{23}^i & P_{24}^i \\ P_{31}^i & P_{32}^i & P_{33}^i & P_{34}^i \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \quad (1)$$

$$K = P_{31}^i * x + P_{32}^i * y + P_{33}^i * z + P_{34}^i \quad (2)$$

For the silhouette of image i must be in its 2D bounding box, we have the following 4 inequations:

$$wl_i \leqslant u = \frac{P_{11}^i * x + P_{12}^i * y + P_{13}^i * z + P_{14}^i}{P_{31}^i * x + P_{32}^i * y + P_{33}^i * z + P_{34}^i} \leqslant wh_i \quad (3)$$

$$hl_i \leqslant v = \frac{P_{21}^i * x + P_{22}^i * y + P_{23}^i * z + P_{24}^i}{P_{31}^i * x + P_{32}^i * y + P_{33}^i * z + P_{34}^i} \leqslant hh_i \quad (4)$$

Hence, the problem needs to solve can be generalized as how to calculate the maximum and minimum of x, y, z under $4N$ linear inequations. We use genetic algorithm to solve this problem. In this algorithm, each individual is a 3D point (x, y, z). If we want to calculate the maximum or the minimum of x, the corresponding object function is $f_{obj} = x$ or $f_{obj} = -x$. The fitness function is equal to the object function. The algorithm is:

- step 1. Generate an initial population that satisfies the $4N$ inequations. If we know the approximate position of the object, the step will be much easier.
- step 2. Select individual according to its fitness.
- step 3. Cross between individuals and mutate. If invalid individual that does not satisfy the $4N$ inequations is generated, it should be discarded.
- step 4. Save the best individual of this generation. If the maximum allowed number of generation is not achieved, go to step (2), else exit.

We take a real object as an example(see Fig. 4), using this algorithm to compute its bounding box. From Table I, we can see that an accurate bounding box can be computed by genetic algorithm.

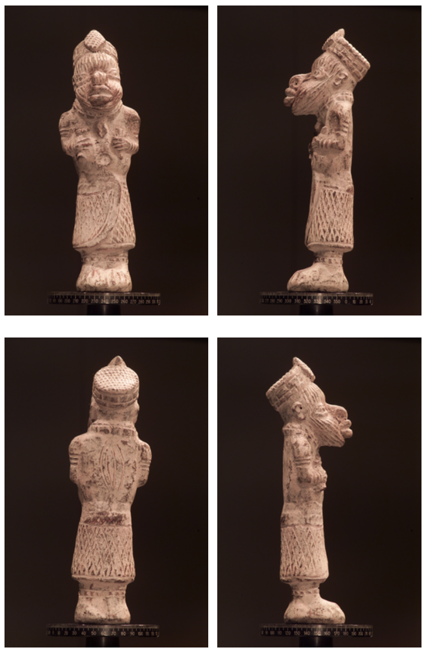

Fig. 4: Four samples of a total of 36 color images (2008x3040 pixels) used in the reconstruction of the BigHead model, courtesy of C. H. Esteban [3].

TABLE I: 3D bounding box of the BigHead object

	Real object	Compute by GA	Object function
X min	-50.1199	-50.0193	$f = -x$
X max	50.6254	50.5407	$f = x$
Y min	-221.9601	-221.1833	$f = -y$
Y max	33.5465	33.4649	$f = y$
Z min	-51.4232	-51.3441	$f = -z$
Z max	57.6349	57.6167	$f = z$

B. Build octree construction of visual hull

An octree of an object can be constructed from the silhouettes through recursive subdivision and projection tests. Starting from the bounding box, the octree subdivides a cube into 8 children whenever it is on the isosurface, and iterates the process recursively until the maximum level of depth is reached. In the case of a visual hull construction, we have to define an isosurface function that corresponds to the visual hull surface. To make the isosurface function value indicate the 3D distance to visual hull surface, compute the 2D distance to the contours of the silhouettes. Then we get a visual hull isovalue function for a given 3D point v as:

$$f_{iso}(v) = \max_i D_i(P_i * v), \quad i = 1, 2, \cdots, N. \quad (5)$$

where D_i is a chamfer distance transform [14] to the contour of silhouette i, negative inside and positive outside the silhouette.

To evaluate a given voxel, we project it into all the silhouettes to assign it one of the 3 available types. In this paper, our projection test method to evaluate a given voxel is: calculate the isofunction value of 8 corners of the voxel.

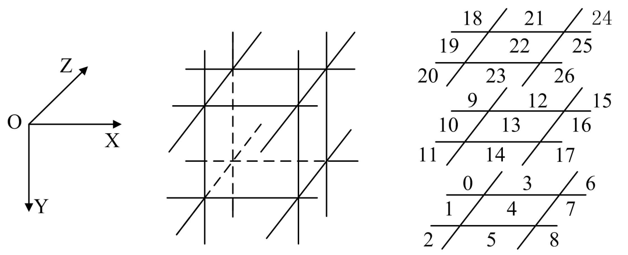

Fig. 5: Twenty seven partitions of a 3D volume by the six planes of a cube.

- When the 8 corners of the voxel are out of the visual hull, project the voxel to all the images. If in one image the projection of the voxel is completely out of the silhouette, then the voxel's type is out. Else, its type is on.
- When the 8 corners of the voxel are in the visual hull, project the voxel to all the images. If in all the images the projection of the voxel is completely in the silhouettes, then the voxel's type is in. Else, its type is on.
- When parts of 8 corners of the voxel are in the visual hull, its type is on.

In fact, we can know the type of a voxel only by justifying the relative position between the projection of the voxel and the silhouettes. Here we use the isofuction value of 8 corners of the cube to decrease computing time. Since when parts of 8 corners of a voxel are in the visual hull, we need not calculate the projection of the cube to know its type is on. On the other hand, we need the isofunction value of corners of the voxel in marching cubes algorithm.

Our projection test method can be realized in two steps. The first step is to compute the projection of a voxel onto all the images, which is not an easy problem. In fact, this problem is solved by Potmesil by justifying the relative position between a cube and the projection center of an image, then getting the projection of the cube based on a lookup table [9]. Although his method is quite intuitive, it need know the position of the projection center relative to the cube and can not be implemented efficiently. This paper presents a new method to calculate the projection of a cube in an image, which only needs 8 corners of the cube and projection matrix of the image. Our method is: project 8 corners of the cube to the image, then calculate the convex hull of projections of 8 corners. In fact, this convex hull is the same as the projection of the cube. Now, we prove it: the six faces of a cube partition the 3D space of the cube into 27 half-spaces: 26 outside and 1 inside the cube (see Figure. 5). The silhouettes of the cube as seen from the twenty six outside partitions are variations on three basic shapes, shown in Fig. 6, made of one, two, or three visible faces. From Fig. 6 we know that the projection of the cube is the same as the convex hull of projections of 8 corners whenever the silhouettes of the cube are variations on one, two, or three visible faces.

We use the gift wrapping algorithm [15] which is a simple

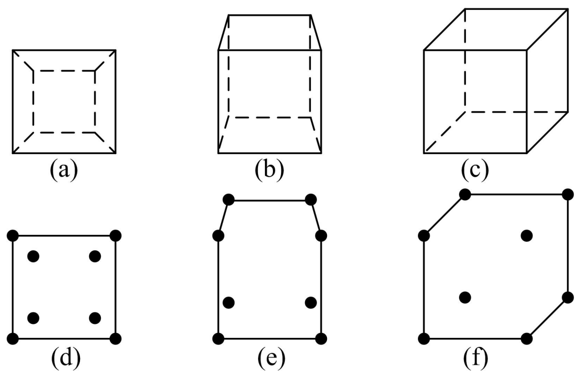

Fig. 6: Three perspective projections of a cube into a plane with (a) one, (b) two, and (c) three visible faces. (d) (e) (f) The convex hull of projections of 8 corners of the cube in corresponding surroundings.

algorithm for computing the convex hull of a given set of points to calculate the projection of a cube in an image. The gift wrapping algorithm has $O(nh)$ time complexity, where n is the number of points and h is the number of points on the convex hull. In our case, n is equal to 8, h is less than or equal to 6. Let p is equal to the projections of 8 vertices of the cube. The gift wrapping algorithm can be coded as:

```
GiftWrapping(p)
i = 0.
p[0]=left most point of P.
do{
    p[i+1]=point. Where all other
    points in P is to the right of the
    line p[i]p[i+1].
    i=i+1.
}while p[i]!=p[0]
return p.
```

After the projection of a cube is computed, the next step is to justify the relative position of the projection of the cube to the silhouettes. Because the projection of the cube is a polygon and the polygon is convex, the scan processor needs to track only the right and the left edges from the top to the bottom of the polygon. If all the pixels in the polygon are inside the object regions, the polygon is in the silhouette. If all the pixels in the polygon are outside the object regions, the polygon is out of the silhouette. If parts of the pixels in the polygon are inside the object regions, the polygon intersects with the silhouette. We can see the octree construction of the BigHead object for different levels of resolution in Fig. 7 computed by our method.

C. *Compute visual hull mesh by marching cubes algorithm*

Once the octree is reconstructed, marching cubes algorithm is applied to extract the visual hull surface. To do this, the voxel occupancy of a leaf node is encoded into 8 values using the isofunction value of its eight vertices. Since we do a distance transform, this isofunction value can represent the 3D distance to visual hull surface, negative inside and

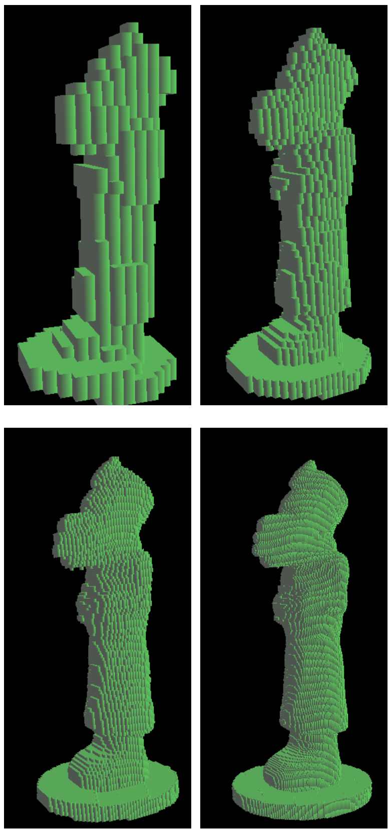

Fig. 7: Octree construction of the BigHead model. From left to right, the level of detail ranges from 5 to 8 depth levels.

positive outside the visual hull surface. These values are then used to index into a pre-defined lookup table which defines surface triangles within the voxel that will form part of the final visual hull mesh. In Fig. 8 we present the visual hull mesh of BigHead object for 8 levels of resolution.

IV. EXPERIMENT RESULTS

To evaluate the contributions of our approach, we demonstrate the reconstructions of several real world objects. The image sequences of these objects are acquired with an electronic turntable. Our system requires accurate segmentation each of the image into silhouette. However, it is difficult to extract silhouette automatically from images with complex background. In our 3D object scanning case, it is desirable to segment the silhouette automatically. There are some variational techniques, like level-set based methods, snake based methods can accomplish this task. However, when the resolution of image is very high, these approaches will be very time consuming and subject to variations in cameras, lighting, and background materials. To facilitate silhouette segmentation, we use a monochrome background in the setup of image acquisition. So it is easy to find the object silhouette using standard background subtraction method. Fig. 9 shows two sets of image sequences of two Terra Cotta Warriors.

Our visual hull computation algorithm is implemented on a Pentium IV 3.0GHz PC with 1.5G RAM and WindowsXP OS. Fig. 10 gives some experimental results to reveal the

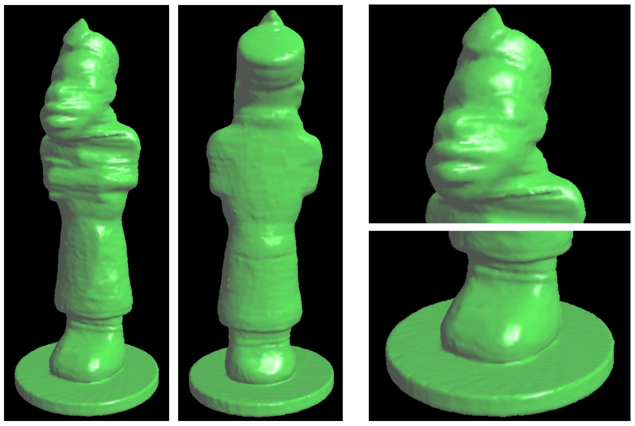

Fig. 8: Marching cubes meshes of the Bighead object for 8 levels of resolution. The right shows two zoomed portion of the mesh model.

TABLE II: Computational parameters of visual hull mesh

	Num. Images	*Resolution*	*Time*	*Triangle*
BigHead	36	2008×3040	7.5min	132970
Minister	36	3088×2056	6.3min	110000
Captain	36	3088×2056	8.6min	152320

visual hull computed from image sequences in Fig. 9. In Table II, we present a portion of the computational times and other parameters of the visual hull construction. Computational times are dominated by the octree construction step. A typical computation time for 36 images of 6M pixels is about 7 minutes with 8 level of octree, while the mesh extract step costs only a few seconds. In Fig. 10, we can see that the proposed approach improves significantly the quality of the reconstructed visual hull relative to the binary MC grid.

V. CONCLUSIONS

In this paper, we have proposed a new volumetric based method to construct high quality visual hull mesh from silhouettes. We also introduce a simple and efficient voxel projection test method in the process of building octree construction to avoid ambiguity. The proposed approach is robust with regards to objects having complicated topology and very easy to implement. On the other hand, our method significantly improves the quality of the reconstructed visual hull. The results show that the algorithm is efficient from both theoretical and practical standpoints. In the future, we will accelerate the algorithm by speeding up the projection test process and efficiently traversing octree. Having high quality visual hull, we can reconstruct 3D mesh with photo-consistence in framework of level set or snake.

VI. ACKNOWLEDGMENTS

This project is partially supported by Natural Science Foundation of China (NSFC No.50805031). Image sequences of Bighead are courtesy of `http://www.tsi.enst.fr/3dmodels/html/enst/bigHead.html`.

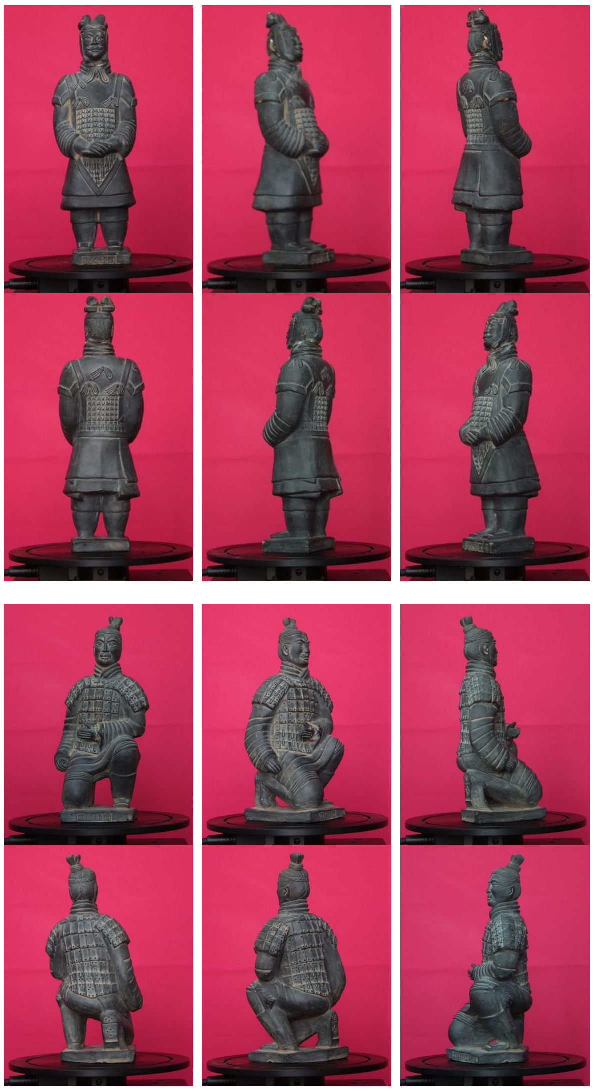

Fig. 9: Top: 6 samples of a total of 36 color images (3088×2056 pixels) of the Terra Cotta Warrior Minister sequence. Bottom: 6 samples of a total of 36 color images (3088×2056 pixels) of the Terra Cotta Warrior Captain sequence.

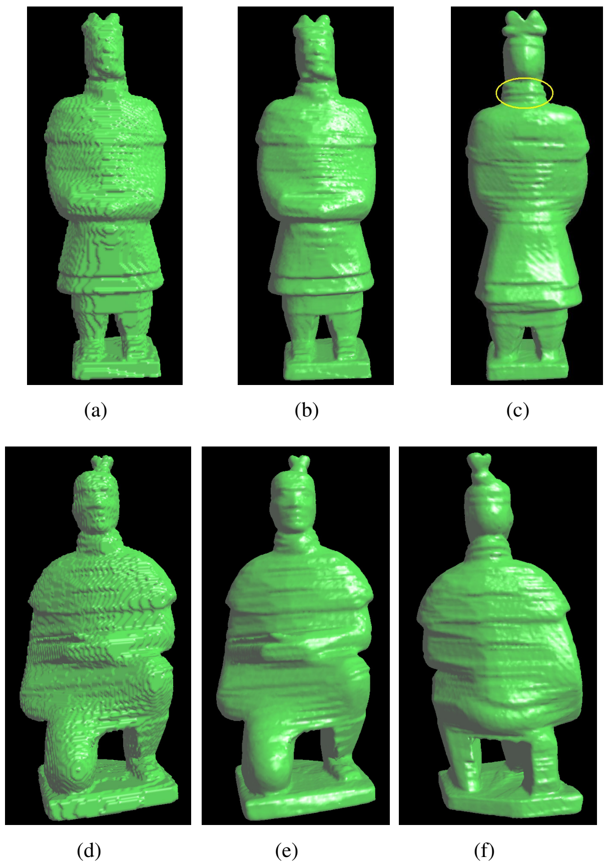

Fig. 10: (a)(b)(c) are visual hull meshes of the Terra Cotta Warrior minister sequences produced by binary MC grid and our proposed approach, respectively. From (c) we can find the fine details in the yellow ellipse can be recovered. (d)(e)(f) are visual hull meshes of the Terra Cotta Warrior Captain sequences produced by binary MC grid and our algorithm.

References

[1] A. Laurentini, "The Visual Hull Concept for Silhouette Based Image Understanding", *IEEE Trans. on PAMI*, vol.16, no.2, 1994, pp. 150-162.

[2] G. Cross and A. Zisserman, "Surface Reconstruction from Multiple Views Using Apparent Contours and Surface Texture", in *NATO Advanced Research Workshop on Conference of Computer Vision and Computer Graphics*, 2000, pp. 25-47.

[3] C. H. Esteban, "Stereo and Silhouette Fusion for 3D Object Modeling from Uncalibrated Images under Circular Motion", *PhD thesis, ENST*, 2004.

[4] B. G. Baumgart, "A Polyhedron Representation for Computer Vision", in *AFIPS National Computer Conferenc Proceedings*, vol.44, 1975, pp. 589-596.

[5] W. Matusik, C. Buehler, and L. McMillan, "Polyhedral Visual Hulls for Real-Time Rendering", in *Proc. 12th Eurographics Workshop on Rendering*, 2001, pp. 115-125.

[6] W. Matusik, C. Buehler, R. Raskar, S. Gortler, and L. McMillan, "Image-based Visual Hulls", in *Proc. SIGGRAPH*, 2000, pp. 369-374.

[7] I. Shlyakhter, M. Rozenoer, J. Dorsey, and S. Teller, "Reconstructing 3D Tree Models from Intrumented Photographs", *IEEE Computer Graphics and Applications*, vol.21, no.3, 2001, pp. 53-61.

[8] R. Szeliski, "Rapid Octree Construction From Image Sequences", in *CVGIP: Image Understanding*, vol.1, no.58, 1993, pp. 23-32.

[9] M. Potmesil, "Generating Octree Models of 3d Objects from Their Silhouettes in a Sequence of Images", *Computer Vision, Graphics, and Image Processing*, vol.40, 1987, pp. 1-29.

[10] W. Martin and J. Aggarwal, "Volumetric Descriptions of Objects from Multiple Views", *IEEE Trans. on PAMI*, vol.5, no.2, 1983, pp. 150-158.

[11] K. M. Cheung, T. Kanade, J. Bouguet, and M. Holler, "A Real Time System for Robust 3D Voxel Reconstruction of Human Motions", in *CVPR*, Vol.2, 2000, pp. 714-720.

[12] W. E. Lorensen and H. E. Cline, "Marching Cubes: A High Resolution 3d Surface Construction Algorithm", in *Proc. SIGGRAPH*, vol. 21, 1987, pp. 163-169.

[13] A. Erol, G. Bebis, R. Boyle, and M. Nicolescu, "Visual Hull Construction Using Adaptive Sampling", in *Proc. 7th IEEE Workshops on Application of Computer Vision*, vol.1, 2005, pp. 234-241.

[14] G. Borgefors, "Distance Transformations in Digital Images", *Computer Vision, Graphics, and Image Processing*, vol.34, 1986, pp. 344-371.

[15] R. A. Jarvis, "On the Identification of the Convex Hull of a Finite Set of Points in the Plane", *Information Processing Letters*, vol.2, 1973, pp. 18-21.