2010

3D Object Reconstruction using Multi-View Calibrated Images

Mohammad R. Raeesi N.
University of Windsor

3D Object Reconstruction using Multi-View Calibrated Images

by

Mohammad R. Raeesi N.

A Thesis
Submitted to the Faculty of Graduate Studies
through Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Master of Applied Science at the
University of Windsor

Windsor, Ontario, Canada

2010

Your file Votre référence
ISBN: 978-0-494-62747-1
Our file Notre référence
ISBN: 978-0-494-62747-1

Author's Declaration of Previous Publication

This thesis includes one original paper that has been previously submitted for publication in peer reviewed conference, as follows:

Thesis Chapter	Full Citation	Publication Status
Chapter 4	Mohammad R. Raeesi N., Q. M. Jonathan Wu, "A Complete Visual Hull Model Using Bounding Edges", Pacific Rim Conference on Multimedia 2010.	Submitted

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as graduate student at the University of Windsor.

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

Abstract

In this study, two models are proposed, one is a visual hull model and another one is a 3D object reconstruction model. The proposed visual hull model, which is based on bounding edge representation, obtains high time performance which makes it to be one of the best methods. The main contribution of the proposed visual hull model is to provide bounding surfaces over the bounding edges, which results a complete triangular surface mesh. Moreover, the proposed visual hull model can be computed over the camera networks distributedly. The second model is a depth map based 3D object reconstruction model which results a watertight triangular surface mesh. The proposed model produces the result with acceptable accuracy as well as high completeness, only using stereo matching and triangulation. The contribution of this model is to playing with the 3D points to find the best reliable ones and fitting a surface over them.

Dedication

This thesis is dedicated to my beloved parents, who have raised me to be the person I am today. You have been with me every step of the way, through good times and bad. Thank you for all the unconditional love, guidance, and support that you have always given me, helping me to succeed and instilling in me the confidence that I am capable of doing anything I put my mind to. Thank you for everything. I love you!

Acknowledgements

I would like to thank all people who have helped and inspired me during my graduate study.

I especially want to thank Dr. Jonathan Wu, my supervisor, for his support and guidance throughout this entire thesis process and for believing in my abilities. Dr. Shervin Erfani and Dr. Nader Zamani deserve special thanks as my thesis committee members. Thank you for your support and guidance.

I am indebted to all of my roommates during past two years for their friendship, care and support. I would like to thank Abbas Ghaei, Amir Hassannejadasl, Ali Aryanpour, and especially Esmaeal Navaei. Good luck to each of you in your future endeavors.

My deepest gratitude goes to my family for their unflagging love and support throughout my life; this dissertation is simply impossible without them. I am indebted to my parents, for their care and love. I cannot ask for more from them, as they are simply perfect. I have no suitable word that can fully describe their everlasting love to me. I remember their constant support when I encountered difficulties. I feel proud of my family, for their support and care in my whole life.

Last but not least, thanks to God for my life through all tests in the past two years. You have made my life more bountiful. May your name be exalted, honored, and glorified.

Table of Contents

Author's Declaration of Previous Publication iii

Abstract iv

Dedication v

Acknowledgements vi

List of Figures x

List of Tables xiii

1. Introduction 1

2. Fundamental Concepts 5

2.1. Distributed Vision Network 5

2.2. Distributed Computing 6

2.3. Silhouettes 6

2.4. Visual Hull 9

2.4.1. Voxel Based Visual Hulls 12

2.4.2. Polyhedral Visual Hulls 14

2.4.3. Image-Based Visual Hulls 17

2.4.4. Bounding Edge Visual Hull 18

2.5. Visual Hull across Time 20

2.6. Comparison 23

2.7. Stereo Vision 25

3. Implementation Platform 28

3.1. Implementation Methodology 28

3.2. Datasets 30

3.3. Silhouette Images Computation 34
4. Proposed Visual Hull Model 38
4.1. The Algorithm 38
4.1.1. Modified Bounding Edge Model 40
4.1.2. Bounding Surfaces 45
4.1.3. Merging Bounding Surfaces 45
4.1.4. Re-Meshing 46
4.2. Experiments 47
4.2.1. Results 47
4.2.2. Comparison and Evaluation 51
4.3. Conclusion 53
5. The Proposed 3D Object Reconstruction Model 54
5.1. Existing Models 54
5.1.1. 3D Volumetric Approaches 57
5.1.2. Surface Evolution Techniques 57
5.1.3. Feature Extraction and Expansion Algorithms 58
5.1.4. Depth Map based Methods 59
5.2. Surface Reconstruction Methods 60
5.3. The Proposed Algorithm 62
5.3.1. Producing Depth Maps 62
5.3.2. Merging 3D Points 67
5.4. Experiments 69
5.4.1. Results 69
5.4.2. Comparison and Evaluation 72
6. Conclusions 78

7. References 80

Appendix A: Implementation using Java 88

Appendix B: Comparison of the 2nd Proposed Model 94

VITA AUCTORIS 98

List of Figures

Figure 1-1. Sample views of DinoSparseRing dataset [2]. 2
Figure 1-2. 3D convex hull for DinoSparseRing dataset. 2
Figure 1-3. Visual hull model of DinoSparseRing dataset. 3
Figure 1-4. Surface points for DinoSparseRing dataset 4
Figure 2-1. Sample view (a) of Dinosaur dataset and its silhouette image (b) [7]. 7
Figure 2-2. The polyhedron representation of a plastic horse produced by Baumgart [6]. 7
Figure 2-3. Sample application of chromakeying. 8
Figure 2-4. Intersection of silhouette cones. 9
Figure 2-5. Two different images with the same silhouette set. 10
Figure 2-6. The division of Laurentini of the object surfaces [12]. 11
Figure 2-7. Reconstructed visual hull based on different number of views [13]. 11
Figure 2-8. A simple octree structure to model the 3D object [16]. 13
Figure 2-9. The synthetic objects and their octree models produced by R. Szeliski [16]. 13
Figure 2-10. Two rims intersecting at a frontier point [19]. 15
Figure 2-11. The silhouette image and the corresponding edge-bin structure [20]. 16
Figure 2-12. The results of the proposed algorithm in [13] on a torus in different number of views with the processing time. 16
Figure 2-13. A sample slice of the image based visual hull [21]. 17
Figure 2-14. The rays and their corresponding projected rays [22]. 18
Figure 2-15. A bounding edge through the first camera [23]. 18
Figure 2-16. A sample view of a 3D object and corresponding bounding edge model from two views [11]. 19
Figure 2-17. A sample view of a 3D object and the corresponding colored surface points from two different views [11]. 21
Figure 2-18. Forward and backward moving of the colored surface points have been shown as well as the corresponding errors [11]. 22

Figure 2-19. Combining silhouette images from different time instances, by moving the center of the cameras backward [11]. 23
Figure 2-20. Sample image pairs for stereo vision [24] 27
Figure 2-21. The ideal depth map for sample pairs [24] 27
Figure 3-1. A sample view of MeshLab application 29
Figure 3-2. Images of Dinosaur dataset. 31
Figure 3-3. Images of Predator dataset. 32
Figure 3-4. DinoSparseRing dataset images. 33
Figure 3-5. Resulted silhouette images for Dinosaur dataset. 35
Figure 3-6. Resulted silhouette images for Predator dataset 36
Figure 3-7. Resulted silhouette images for DinoSparseRing dataset; sample images (1st row), results of providers suggestion (2nd row), manually refined results (3rd row) 37
Figure 4-1. (a) The bounding edge model for the 1st view of Dinosaur dataset and (b) corresponding contour map 42
Figure 4-2. 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). 44
Figure 4-3. Intersection of the occupancy intervals form different viewpoints 46
Figure 4-4. Bounding surfaces resulted for the first 8 views of Dinosaur dataset 48
Figure 4-5. Bounding surfaces resulted for the first 8 views of Predator dataset. 48
Figure 4-6. Bounding surfaces resulted for the first 8 views of DinSparseRing dataset. . 49
Figure 4-7. Final triangular meshes for Dinosaur datasets. 49
Figure 4-8. Final triangular meshes for Predator datasets. 50
Figure 4-9. Final triangular meshes for DinoSparseRing dataset. 50
Figure 5-1. Ground truth for DinoSparseRing dataset [2]. 56
Figure 5-2. Dinosaur sample image pair 63
Figure 5-3. Rectified Dinosaur image pair. 63
Figure 5-4. Rectified silhouette image pairs. 64
Figure 5-5. Left-to-right (left image) and right-to-left (right image) depth maps produced for the sample image pairs of Dinosaur dataset. 65

Figure 5-6. Left-to-right depth map (left image) and resulted depth map after LRC check (right image) for the sample image pairs of Dinosaur dataset. 66
Figure 5-7. Reconstructed 3D points for the first view of Dinosaur dataset. 66
Figure 5-8. Refined 3D points for the first view of Dinosaur dataset..... 68
Figure 5-9. Resulted 3D surface points for Dinosaur dataset. 70
Figure 5-10. Resulted 3D surface points for DinoSparseRing dataset. 70
Figure 5-11. The inconsistent image of DinoSparseRing dataset..... 71
Figure 5-12. Final 3D triangular mesh for DinoSparseRing dataset. 71
Figure 5-13. Final 3D triangular mesh for DinoSparseRing dataset from different views. 72
Figure B-1. Accuracy comparison among all the state-of-the-art methods. 94
Figure B-2. Completeness comparison among all the state-of-the-art methods. 95
Figure B-3. Accuracy comparison among the depth map based methods. 95
Figure B-4. Completeness comparison among the depth map based methods. 96
Figure B-5. Accuracy comparison among the methods which are not published yet..... 96
Figure B-6. Completeness comparison among the methods which are not published yet. 97

List of Tables

Table 2-1. The comparison of different visual hull models. 24
Table 4-1. Numbers of inconsistent points for the first 8 views of the Dinosaur dataset. 43
Table 4-2. Number of vertices in the all surfaces versus the merged surfaces before re-meshing. 51
Table 4-3. Execution time of the final visual hull model produced by different models in second. 51
Table 4-4. Execution time sequentially versus distributedly in second. 53
Table 5-1. Evaluation results of both proposed models on Middlebury benchmark for DinoSparseRing dataset. The accuracy is in millimeter and completeness is percentage.73
Table 5-2. Comparison of accuracy of the state-of-the-art models. The accuracy threshold is 90% 74
Table 5-3. Comparison of completeness of the state-of-the-art models. The completeness threshold is 1.25mm. 75
Table 5-4. Comparison of accuracy of the state-of-the-art depth map based models. The accuracy threshold is 80%. 76
Table 5-5. Comparison of completeness of the state-of-the-art depth map based models. The completeness threshold is 1.5*mm*. 77
Table A-1. Class files implemented in Java with number of physical and logical lines of code as well as number of implemented methods. 89

1. Introduction

There are many applications such as obstacle avoidance in robotics, 3D modeling in inverse engineering, assisted living, security and surveillance which need to localize, recognize, reconstruct and track the 3D objects. There are many approaches for these applications, such as marker-based tracking which attaches some markers to the interesting objects. Some of the existing approaches are not applicable in many environments; for example, it is not possible to use the marker-based approaches for surveillance applications in public places. The best, applicable approach is vision network, because it is relatively cheaper, and it can be configured easily [1].

The area of these applications is quite wide, including Electrical Engineering, Computer Science, Mechanical Engineering, Medicine, and Security. The goal of all of these applications is to reconstruct the 3D object, but each of them needs the geometric information of the 3D object in different level of details. In obstacle avoidance applications in robotics, for example, moving robot gets the information from the environment using its sensors, and based on the received information it chooses a path to reach the destination without hitting the existing obstacles. Since the robot only needs the location and the course information of the shape of the 3D objects just to move along the objects, there is no need to recover the exact shape of the 3D object. Controversially, in 3D modeling for inverse engineering, the shape of the object and all geometrical information of it should be reconstructed as accurate as possible. Considering the processing time, the reconstruction should be real time for moving robots, while there is no limitation for 3D modeling applications.

In vision networks, there are different algorithms which will result in different 3D reconstructed shape of the object, from a coarse model to the most precise one. The applications in this field recover the 3D shape of the objects based on the captured images from different views of the object. Most of them use the silhouette images to do so. All the applications in this field have the three steps including getting geometrical

information from each image, computing a model of the objects in the scene, representing the objects and making decision about the situation of the objects in the scene.

The inputs used for vision network applications are multi-view calibrated images. Camera calibration is a part of vision network applications which is out of the scope of this study. Figure 1-1 shows sample images from DinoSparseRing dataset [2].

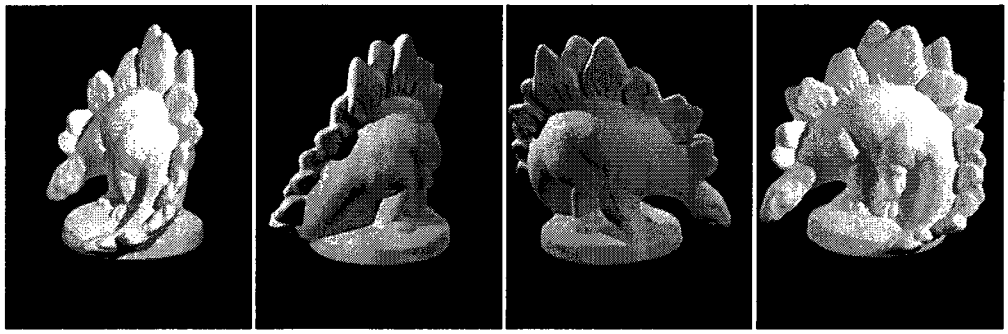


Figure 1-1. Sample views of DinoSparseRing dataset [2].

The coarsest model in camera networks is 3D convex hull. The 3D convex hull of a set of 3D points is the smallest subset of the space such that for any two points u and v, the segment joining them is completely in the subset. Consider DinoSparseRing dataset, for example, the resulted 3D convex hull has been shown in Figure 1-2.

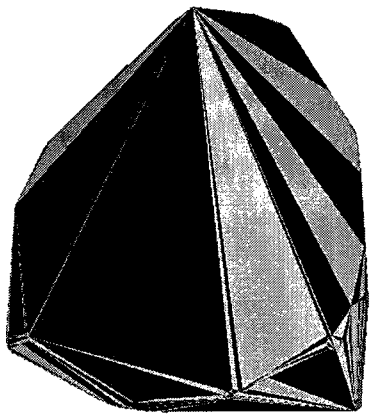


Figure 1-2. 3D convex hull for DinoSparseRing dataset.

A better 3D model for the reconstructed object is visual hull model which is described in the next section as a fundamental concept in 3D reconstruction field. As an overall view, visual hull is the best approximation of the object based on the binary images of the object without any color information. Figure 1-3 shows the resulted visual model for the DinoSparseRing dataset. As it can be seen clearly, the visual hull model is more precise than the convex hull. In other words, visual hull is much more similar to the 3D object than the convex hull model.

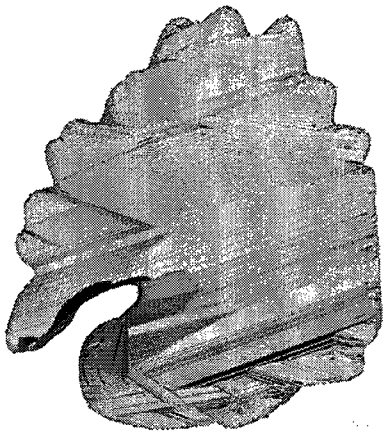


Figure 1-3. Visual hull model of DinoSparseRing dataset.

The 3D reconstructed object is the name of the best model for representing an interesting 3D object, which shows all the concavities of the 3D objects. This model represents all geometric information of the object as accurate as possible. To produce 3D reconstructed object, all the information captured by the cameras will be used including color information. A sample view of the best reconstructed surface points of DinoSparseRing dataset has been shown in Figure 1-4.

The 3D reconstructed object model is the most similar model to the 3D object. As providing more detailed information needs more processing, the execution time of the 3D reconstructed object is much higher than the visual hull and convex hull models.

Figure 1-4. Surface points for DinoSparseRing dataset

In this thesis, I propose two new algorithms, one is a visual hull model and the other one is a 3D reconstructed object model. The proposed visual hull model produces a complete triangular mesh based on the bounding edge model, which is the fastest visual hull model in the existing approaches. The time performance of the proposed model is better than the most existing approaches which provide the same type of results. For the second model, I used depth maps to reconstruct the 3D surface points of the object. To have the reliable depth map, I did a survey in stereo vision to select the best way to do so.

Before describing the proposed models, fundamental concepts are reviewed in section 2. Implementation platform including programming methodology and datasets are mentioned in section 3. Section 4 describes the proposed visual hull model and the obtained results and evaluation. The 3D reconstructed object model is described in section 5, followed by the conclusions in section 6. The last part, Appendix A, describes the implemented codes in Java programming language.

I used Matlab and Java programming languages to implement the codes of the proposed algorithms. Because Matlab is much faster than Java for matrix manipulation, I used Matlab for image processing tasks, such as window matching. For the 3D computation, Java is used which is faster than Matlab in this case. The Matlab version used is 7.0.0.19920(R14), and the Java version is 1.6.0_12.

2. Fundamental Concepts

W. N. Martin and J. K. Aggarwal [3] first described the volumetric description from multiple views. Later, other researches defined the fundamental concepts of the 3D model approximation, such as silhouette and visual hull. These concepts are used in all of the corresponding algorithms.

To provide the multi-view calibrated images, as the input for 3D object reconstruction, there are two approaches. The first one is to use a turntable to rotate the object and a camera to capture images. The second approach is to configure a camera network on the environment under study. If a processing unit is available for the camera nodes in the network, the computation of the algorithms can be distributed over the network. Otherwise, there is a server which processes the captured images from different cameras.

2.1. Distributed Vision Network

The most significant concept to be defined is the Distributed Vision Network. In this study, the definition of the Distributed Vision Networks is the same as the definition of A. Mavrinac [4]. Distributed Vision Networks are networks of dispersed camera nodes. Each node has (i) a camera module for image acquisition, (ii) a processor to process the raw image locally, and (iii) a communication module to send and receive information. This type of network can either use a central device to perform collective processing of the data or perform the processing collaboratively by the nodes. The cameras are calibrated over the network. Camera calibration, also called *camera resectioning*, is the process of finding the true parameters of the cameras that produced a given photograph or video. Camera parameters include the focal length, point of view, global position, global direction, and global rotation. Camera calibration may be done automatically over the network, or as a preprocess step in network configuration.

The field of distributed vision networks is a new and growing field, which is still in the beginning stages of research. This field is a combination of several fields including computer vision, image processing, distributed computing, embedded systems, data networks and communications. This combination adds new opportunities from the union of the fields and new limitations imposed by their intersections [4].

It this study, I consider distributed computing as well as centralized one. The performance of the proposed visual hull model is evaluated on the both types of networks. For the 3D reconstructed object model, all the computations are done sequentially on a centralized server. In this case, a set of camera stations send their captured images to the server and server reconstructs the 3D shape of the objects based on the received images and camera parameters.

2.2. Distributed Computing

Using the distributed computing network environment is beneficial for the Vision Networks in some ways. Distributed computing makes the networks to be scalable. It avoids transmitting the raw images, which have so huge amount of data. In addition, it can preserve the privacy of the network users in some applications such as assisted living. Also, it enhances the flexibility on the type of feature and level of exchanging data. So we will have the fusion across the three dimensions; 3D space (different camera views), time (collecting the data over time) and feature levels (selecting and fusing different feature subsets) [5]. As mentioned before, I consider distributed computing only for the evaluation of the proposed visual hull model. In this case, the first step of the algorithm is computed over the camera nodes in parallel, while the next step which is the merging step is done on a centralized server.

2.3. Silhouettes

All 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. The scene background pixels are most often colored as white, while the foreground pixels are colored as black. These foreground pixels are related to the interesting objects in the scene. Baumgart [6] first considered silhouettes to approximate a polyhedron representation of the objects. He called his work as inverse computer vision, because computer vision generates synthetic images from the real world, while 3D object reconstruction uses the captured images to reconstruct the real object with all the geometric information. A sample silhouette image with its corresponding image has been shown in Figure 2-1.

(a) (b)

Figure 2-1. Sample view (a) of Dinosaur dataset and its silhouette image (b) [7].

Baumgart [6] used three captured images from a plastic horse on a turntable to draw the silhouette images. Then by using the silhouette cone intersection, he produced a polyhedron model of the object. He mentioned that silhouette cone intersection looks like carving a statue by cutting away everything not related to the object. Figure 2-2 shows the 3D reconstructed object using Baumgart techniques. His result polyhedron looks like a statue of a horse which is not completed yet. It seems to be cut by knife.

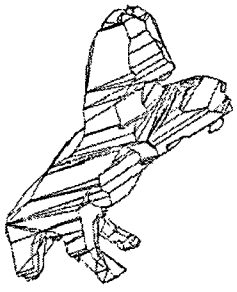


Figure 2-2. The polyhedron representation of a plastic horse produced by Baumgart [6].

There are two common ways to produce the silhouette from the captured images which include *chromakeying* and *background subtraction*. However, in some recent work, the silhouettes are computed manually using Adobe Photoshop, just to segmentation of the foreground and background pixels.

The first approach, chromakeying, also called *bluescreen matting*. In this approach the background is a single uniform color which does not appear in the foreground objects. So by checking the color and compare it with the background color, it is possible to compute the silhouette of the object. This method can not be used in many applications, because of its limitation on the background color, but it is applicable in cinematic special effect and television weather forecasts [8]. A sample application of bluescreen matting has been shown in Figure 2-3. The selected color for the background is green, while there is no green pixel for the foreground object. So only by a comparison of the color of the pixels, it is possible to detect the background pixels and change their value to provide a special effect.

Figure 2-3. Sample application of chromakeying.

Another common way is background subtraction. In this method, first the statistical model of the background is produced by capture many images from the background. So it is possible to detect the foreground objects by comparing the new image with the statistical model of the background. If the difference for any pixel of the

new image is greater than the corresponding threshold, that pixel will be considered as a foreground pixel [9].

For some datasets, the dataset providers provide the contour information of the silhouettes as well as the image data. The contour information is a set of pixels which are not connected. The connected version of these pixels represents the silhouette contour. I implement a function to produce the best connected version of these pixels. Then based on the resulted silhouette contours, the silhouette image is produced. Other datasets suggest the best way to produce the silhouette information. These methods will be described later.

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 silhouette image with the same resolution (without any compression) is less than 8KB.

2.4. Visual Hull

The constructed objects of the silhouettes is called *visual hull*. Visual hull concept was first defined by A. Laurentini [10]. Visual hull is the intersection of the silhouette cones, which are the cones started from the camera positions and goes through the silhouette contours. Figure 2-4 shows the silhouette cones from different viewpoints with different colors. The intersection of all the silhouette cones is called visual hull model of the object.

Figure 2-4. Intersection of silhouette cones.

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 set of silhouettes as the given one. In other word, it is possible for many objects to have the same set of silhouettes; visual hull represents the maximal object. So, it is not possible to identify the objects only based on the silhouette, especially for the non-convex objects. Figure 2-5 shows two different objects which have the same set of silhouettes. So based on the silhouette information, there is no way to recognize any of them.

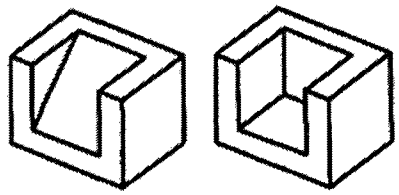


Figure 2-5. Two different images with the same silhouette set.

The visual hull applications and the resulted models are very sensitive to silhouette noise and camera calibration errors.

G. Cheung et al. [11] 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 volume that exactly explains all the silhouette images. Because there are many objects that have the same set of silhouettes, G. Cheung defined the visual hull as the largest possible volume which *exactly explains* the silhouette images.

A. Laurentini [12] divided the surfaces of a volume into two categories, silhouette-active surface and silhouette-inactive surface. The former is what can be reconstructed by the silhouette cone intersection, while the latter one is what can have any shape without affecting the silhouettes of the object. The following figure shows an example of this division. Figure 2-6(a) and (b) shows the two categories of surfaces. The shape of the object in the pentahedron P can not be identified only based on the silhouette

images. So the resulted visual hull in the best situation is what has been shown in Figure 2-6(c), while the real object is one has been shown in Figure 2-6(d).

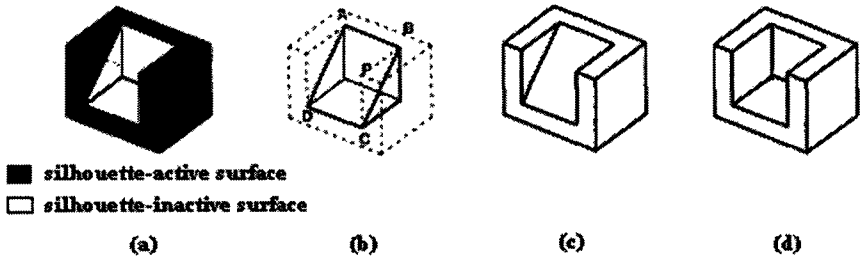


Figure 2-6. The division of Laurentini of the object surfaces [12].

The accuracy of the visual hull mainly depends on the number of silhouette images and their corresponding camera positions. The visual hull will be tighter if the number of silhouette images is increased. The greater the number of the views, the more precise the approximated visual hull. Figure 2-7 shows different reconstructed visual hulls based on different number of views. It also shows the execution time of reconstruction for each set. The execution time is increased by increasing number of views more rapidly.

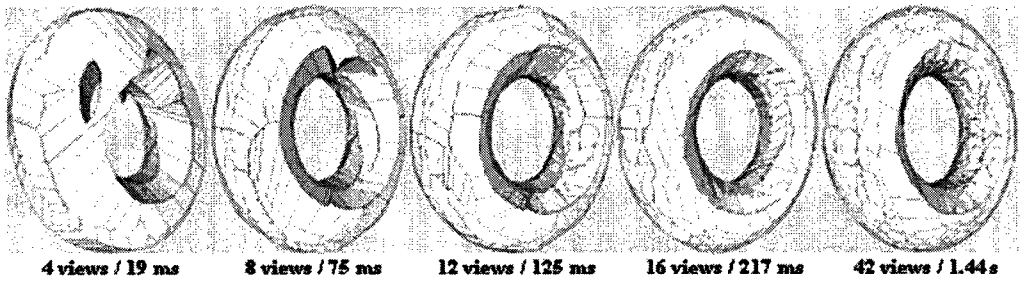


Figure 2-7. Reconstructed visual hull based on different number of views [13].

Based on the survey I did in this field, there are four main categories of visual hull models. In existing approaches for modeling the visual hull, two categories are popular, voxel based approaches and surface based (polyhedron) ones. Other categories are image-based visual hull and bounding edge visual hull. Main existing models are described in the following subsections.

2.4.1. Voxel Based Visual Hulls

The first category is modeling the objects by a collection of elementary 3D cells. These cells are called *voxels* (volumetric pixels), which are first introduced by W. N. Martin and J. K. Aggarwal [3]. Voxels are classified into two categories, the inside and the outside one. If voxels are positioned completely outside the visual hull, in other word, if they have not any intersection at least with one silhouette, they will be classified as outside voxels. Otherwise, if they intersected partially or completely with all the silhouette images, they will be classified as inside voxels. Because voxel based model uses the discrete volumetric representation, it generates some quantization and aliasing artifacts on the resulting model.

The voxel based approaches improved by introducing *octrees*. Octrees have been first introduced by C. L. Jackins and S. L. Tanimoto as an efficient geometric representation [14]. Then octrees have been used for modeling the objects from three orthographic projections by C. H. Chien and J. K. Aggarwal [15].

Octrees are tree-structured representations which are used to model the volumetric data. The octree is constructed by recursively dividing each cube to eight sub-cubes to cover the interesting volume as accurate as possible. There are three possible locations for the cubes including inside the volume, outside the volume and on the boundary. If a cube is completely inside the volume, it will be labeled as inside and its color will be black. If it is completely outside the volume, it will be labeled as outside and it will be colored as white. Otherwise, it will be labeled as boundary and be colored as gray. Based on the application, the gray cubes will be recursively divided to reach the desired accuracy for modeling the object. Figure 2-8 shows the structure of a sample octree model [16].

Figure 2-8 represents an octree model by 6 cubes of two consecutive levels. Each cube has a color, and the gray cubes of the first level has been divided their sub-cubes.

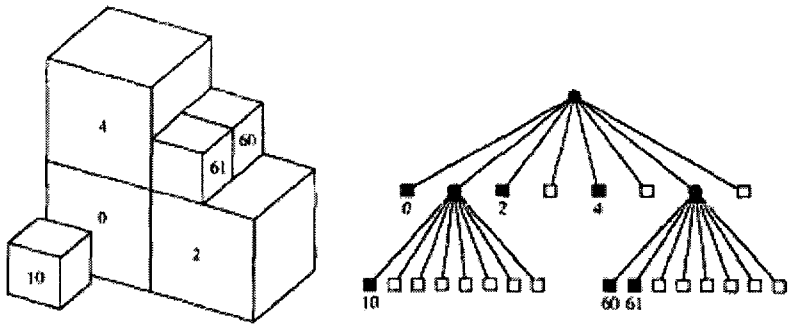


Figure 2-8. A simple octree structure to model the 3D object [16].

R. Szeliski used the octree representation to model the objects. To have an efficient algorithm, he first produced a coarse model of the objects and then by dividing the gray cubes, he refined the model. By using this approach, the number of the trimmed cubes was decreased. To check the location of the cube, it is necessary to project the cube to each image plane to check whether it is inside the silhouette or not. The simplest way is to project the corners of the cube to the image plane and then check the situation of the projected hexagon against the silhouette. This method is accurate, but it is very time consuming. R. Szeliski proposed to convert the cube to a bounding square instead of project its corners. The resulted reconstructed visual hulls of his model have been shown in Figure 2-9 [16].

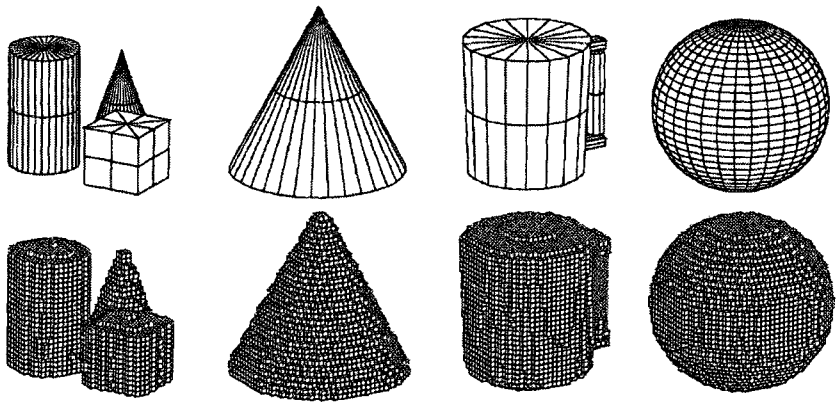


Figure 2-9. The synthetic objects and their octree models produced by R. Szeliski [16].

The octree model performs voxel model. With the same storage space, the precision of the octree model is better than the voxel one. In terms of processing time, the

time needed to construct a voxel model is greater than what the octree model needs, because it evaluates more geometric cells. For representing the reconstructed 3D shape of the object, octree model is also faster, since the number of its geometric cells is less than voxel model.

K. Kutulakos and S. Seitz [17] proposed a voxel based algorithm to model the visual hull. They called it space carving. Space carving algorithm starts on the initial volume and recursively checks the surface voxels to decide whether to carve them or not. It continues checking until no voxels is carved in an iteration.

If the silhouette images are noise free, the smaller voxel size results the better approximation of the visual hull. Otherwise, if the silhouette images are noisy, smaller voxel size causes more errors in classification. So, the best size of the voxels is highly dependent to the error of silhouette images [18].

2.4.2. Polyhedral Visual Hulls

The second category of the popular visual hull modeling 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, the edges of it are the intersection curve between two silhouette cone, and the vertices are the points where more than two silhouette cones intersect. To generalize the polyhedron, it is assumed that the contour of the object has been oriented counterclockwise, so the object is always at the left of the contour.

S. Lazebnik et al. [19] proposed two representations for the visual hull of a 3D object, the *rim* mesh and the visual hull one. Rim is the surface points of the object where a ray through the viewing point intersects the object. The projection of the rim to the corresponding image plane is the silhouette contour. The intersection of two rims could be isolated points which are called *frontier points*. The rim concept has been shown in the following image. 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). Figure 2-10 shows two rims from two different points of view which intersect at a frontier point.

Besides, S. Lazebnik et al. [19] described the difference between the rim mesh and the visual hull one. Because the rim mesh depends only on the ordering of the frontier points, it is topologically more stable. While the visual hull meshes recover the geometry information in a more reliable manner.

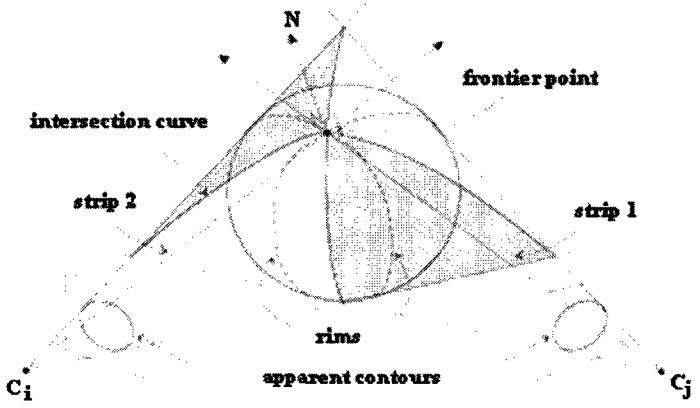
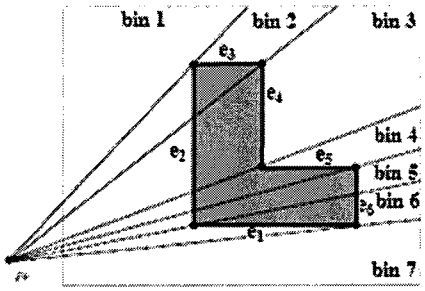


Figure 2-10. Two rims intersecting at a frontier point [19].

C. Buehler et al. [20] proposed the real time representation of the polyhedral visual hulls. Their representation is view independent, so it does not need to be reproduced for a set of silhouette images. It is suited to be computed by the graphics hardware. They assumed each silhouette is a 2D polygon. For each edge of the polygon, they compute the face of the silhouette cone. By using the intersection of the face and other silhouette images, the face of the polyhedron is determined which itself is a polygon. To intersect the face of the silhouette cone by the other silhouette images, the edges of the face of the silhouette cone have been projected to the silhouette images. To accelerate this process, preprocess has been done for each silhouette images. In the preprocessing step, each silhouette has been divided to the bins. Based on these bins, a table of the edges-bins is computed. Figure 2-11 shows the divided silhouette image and the corresponding edge-bins table. The algorithm uses this table to respond as quickly as possible to the intersection problem. This edge-bin structure can be used for the visibility issue as well.

Figure 2-11 shows an example of an edge-bin structure. The silhouette image is divided into seven bins. The cells of the edge-bin table contain the intersecting edges which are sorted ascendingly, based on the distance to the epipole increasingly.



Bin	1	2	3	4	5	6	7
Edges	$\emptyset$	e_2, e_3	e_2, e_4	e_2, e_5	e_2, e_6	e_1, e_6	$\emptyset$

Figure 2-11. The silhouette image and the corresponding edge-bin structure [20].

J. S. Franco and E. Boyer [13] 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, they generated the surfaces of the mesh. Finally, they identified the faces of the polyhedron. They applied their algorithm on a torus for different number of views. The greater the number of the views, the more precise the approximated visual hull. Their results have been shown in Figure 2-12. The time needed to compute each result has been shown as well, which is increased faster than the number of views.

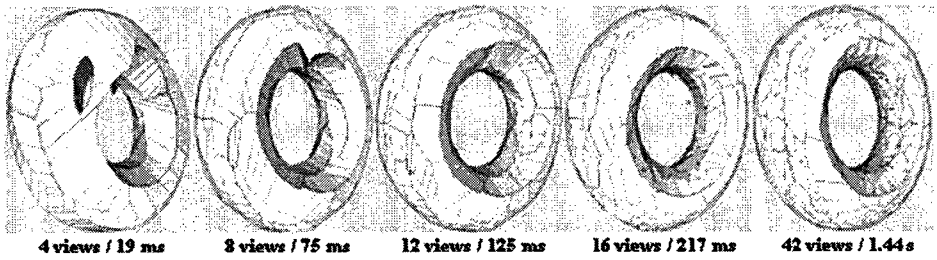


Figure 2-12. The results of the proposed algorithm in [13] on a torus in different number of views with the processing time.

2.4.3. Image-Based Visual Hulls

On of the compact representations of the visual hull is *image-based* visual hull. C. Buehler et al. [21] defined the image-based representation as a two dimensional, sampled representation. For example, a color image is a 2D color samples, or a disparity map in stereo vision is a 2D disparity samples.

The image-based visual hull is a two dimensional, occupancy intervals samples. The visual hull is represented by the rays from view points through the image plane. Instead of storing either foreground or background, 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. If a pixel is a background pixel, its list will be empty. Figure 2-13 shows a slice of the image-based visual hull [21].

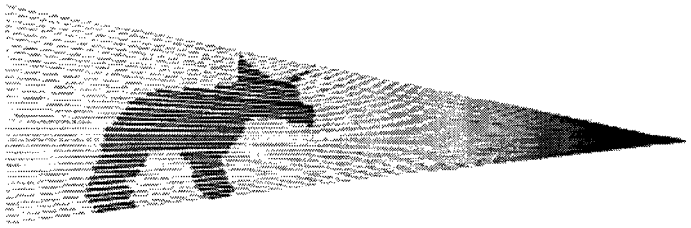


Figure 2-13. A sample slice of the image based visual hull [21].

The image-based representation has many advantages in comparison to other models. Its storage requirements and computational complexity are very low, which is much less than the aforementioned algorithms. It has a simple and fast computation. The rendering is too simple as well. Because it has two discrete dimensions and one continuous dimension, it has a higher resolution than the resolution of the voxel based representation.

W. Matusik et al. [22] proposed an image-based approach to represent the visual hull. Based on the *Calculatus Eliminatus* principle, they said that the visual hull approximation is carving away the regions of space where the object is not. *Calculatus Eliminatus* principle states that we should look everywhere that an object is not located,

and it must be in the place we haven't looked. Their algorithm contains three steps. It first projects the rays from a desire viewing point to the silhouette images. Then, in the second step, by intersecting the projected rays and the silhouettes, it computes the interesting intervals. Finally, the intervals are returned to the 3D space. The intersected parts of these intervals from all the silhouettes will be saved for each pixel. A plot of the projected rays is shown as follows. Figure 2-14 shows the rays from the point of view of an image through the interesting pixels and their corresponding projected lines on the plane of another image. All the projected lines intersect at the epipole, the projected point of the corresponding point of view.

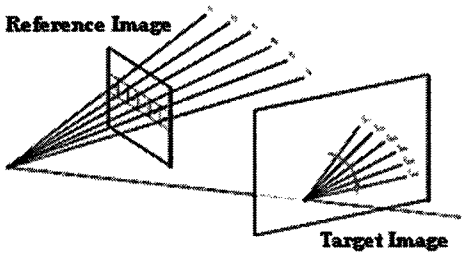


Figure 2-14. The rays and their corresponding projected rays [22].

2.4.4. Bounding Edge Visual Hull

The next visual hull representation is *bounding edges*. This representation has been first introduced by G. Cheung [18, 23, 11, 1]. He defined the bounding edge representation as the parts of the rays from the viewing point through the contour of the corresponding silhouette image which intersect all the other silhouette images.

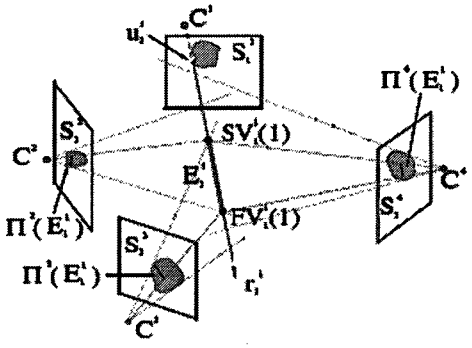


Figure 2-15. A bounding edge through the first camera [23].

Figure 2-15 shows a bounding edge of first camera for a sample pixel which is located on the 3D ray from the camera position through the pixel, and is intersected with other three silhouettes. The starting point *SV* and finishing point *FV* will be saved for the pixel as the endpoints of a bounding edge. To have a better performance, instead of saving the 3D position of each endpoint, its distance to the camera position will be saved [23].

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. However, in contrast to image-based representation, it considers only the silhouette contour pixels instead of all the silhouette pixels. Image based representation is view dependent, since it produces the interval samples for only one view, but bounding edge model produces the edges for all viewpoints. An example of bounding edge representation has been shown in Figure 2-16 [11].

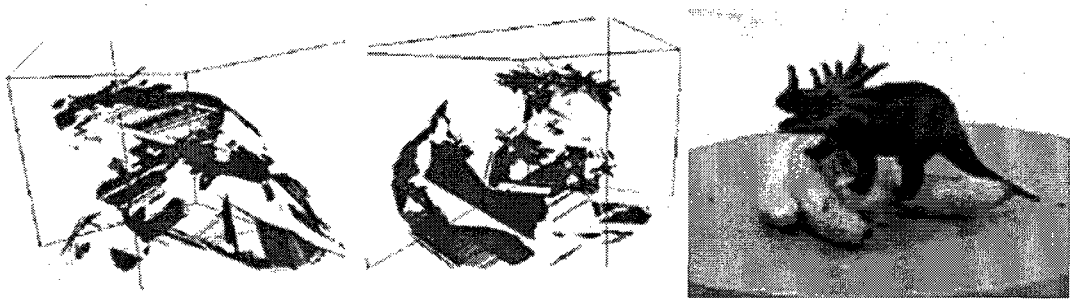


Figure 2-16. A sample view of a 3D object and corresponding bounding edge model from two views [11].

Bounding edges lie exactly on the surface of the visual hull. They intersect the real object at least at one point. The drawback of this representation is that it is incomplete. A visual hull is complete, if it has all the geometrical information of its shape. If visual hull has any holes on its surface, it will be considered as incomplete model. Since the small amount of accurate data is better than the large amount of the approximated data, in many applications, an exact visual hull is preferred than the complete one.

It is not necessary for the bounding edges to be continuous. They can consist of many parts. It exactly depends on the shape of the silhouette images. If all the silhouette images are convex, the bounding edges will be continuous. The important usage of the bounding edge representation is refinement of the visual hull across time. G. Cheung et al. [11] proposed an algorithm to increase the number of silhouettes by capturing more images of the interesting objects over the time.

2.5. Visual Hull across Time

As mentioned before, the visual hull accuracy depends mainly on the number of silhouette images. To increase the precision of the visual hull, the number of silhouette images should be increased. There are two ways to increase the silhouettes number. The first one is to increase the number of cameras to produce more silhouette images which has the financial cost and the limitation for the camera positions. The second approach is using the same cameras across time. G. Cheung et al. [11] proposed an algorithm to increase the number of silhouette by capturing more images of the interesting objects over the time. If there are K cameras in the environment, and J frames of each camera are used, the effective number of silhouette images will be JK instead of K. To apply this approach, first it is necessary to calculate the motion of the interesting object between the time instances. Then silhouette images in different time instances can be combined based on the computed motion over the time. The task of estimating the motion of the object is called *visual hull alignment*, and the task of combining the silhouette images is called *visual hull refinement*.

G. Cheung et al. [23, and 11] considered two fundamental properties of the visual hull. The first one, called 1^{st} *FPVH* (First Fundamental Property of Visual Hull), is that the 3D object which produces the silhouette images lies completely inside the visual hull. This property can be used for many applications such as obstacle avoidance in robotic navigation. The second one, called 2^{nd} *FPVH*, is that each bounding edge touches the real object at least at one point. This property is very important, such that it makes the

construction of the *colored surface points (CSP)* possible. To construct the colored surface points, they combined the 2nd FPVH with the stereo vision concept.

Colored surface points have been first introduced by G. Cheung [18]. CSPs are the 3D points on the bounding edge which touch the object. The information needed to find the position of the CSPs through the bounding edge comes from the color consistency check algorithm. It finds the best point on the bounding edge, the color of which is consistent in all other color images for which the point is visible. The visibility is another important issue to refine the visual hull across time. Because at least one point exists on the bounding edge to touch the object, there is no need to define any threshold. An error measurement algorithm is applied to find the best point. This error is the variance of the color of the point from different visible point of view. In noise free environment, the color variance of a 3D point in all the points of view should be zero. But in real environment, it is necessary to find the best point with least color variance. Figure 2-17 represents the visual hull by its colored surface points. These CSPs are on the surface of the visual hull. In contrast to all other representation, CSPs are colored 3D points.

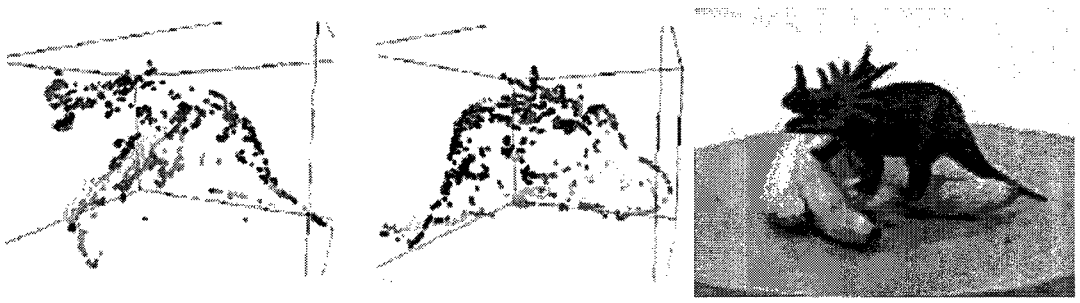


Figure 2-17. A sample view of a 3D object and the corresponding colored surface points from two different views [11].

To estimate the motion of the object, G. Cheung et al. [11] used the 3D CSPs. The algorithm starts with an initial motion, followed by the refinement step. To decrease the complexity of the algorithm, CSPs are moved forward from the first time instance to the next one based on the initial motion, and then projected to the images in the second time instance which is followed by the 3D consistency check. This approach is done in inverse direction; the 3D CSPs from the second time instance are moved backward to project to

the silhouettes of first time instance. Figure 2-18 shows the forward and backward movement of the 3D CSPs through the time instances.

To refining the motion, G. Cheung et al. [11] used two error measurements including *photometric* error and *geometric* one in both directions, forward and backward. The forward photometric error is the color difference between the CSPs in the first time instance and the projected moved points in the second time instance. If the moved points project in the background section of the corresponding silhouette, the photometric error will be considered as zero. Because the images may be noisy and there are some errors in camera calibration in practical cases, they considered valid. For these cases, geometric error is defined, which is the distance of the projected moved CSPs to the silhouettes. This error is zero, when the points project inside the silhouette [11].

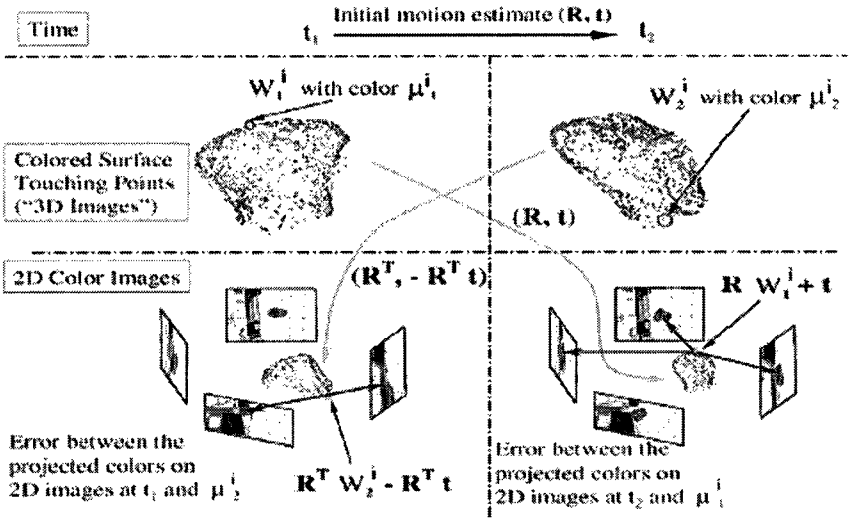


Figure 2-18. Forward and backward moving of the colored surface points have been shown as well as the corresponding errors [11].

After finding the best approximation of the motion, it's time to combine the silhouette images, called visual hull refinement. To do so, first a reference time instance should be set. The first time instance is the best candidate for the reference. The silhouette images from other time instances are effective silhouette images in reference time instance, whose points of view are moved, based on the corresponding refined motion. Figure 2-19 shows the visual hull refinement algorithm.

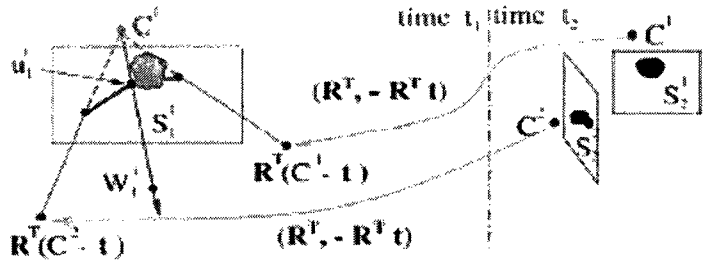


Figure 2-19. Combining silhouette images from different time instances, by moving the center of the cameras backward [11].

G. Cheung et al. [11] applied the proposed algorithm on the rigid objects as well as the articulated objects. The articulated objects are the objects that have many parts which can move in different directions. The best example of an articulated object is human body. The joint estimation is separated from the motion estimation to decrease the complexity of the algorithm. The shape and the motion of all the object parts are computed individually, and then the position of the joints are localized.

2.6. Comparison

The four different representations of visual hull have been described. The entities of the representations are different. The voxel-based one represents the visual hull by the cube cells, while the polyhedral representation models it by the faces, edges and vertices of the polyhedron. The image-based visual hull consists of a two dimensional interval samples, and the bounding edge model represents the visual hull by the bounding edges from different viewpoint.

The comparison of the visual hull models has been shown in Table 2-1. Two factors of the comparison should be defined here which are completeness and exactness. As it mentioned before, a visual hull is complete, if it has all the geometrical information of its shape. If visual hull has any holes on its surfaces, it will be considered as incomplete model [18]. Exactness is a term which refers to the quantization and discretization issues. If a visual hull model uses any type of quantization, it will be considered as an inexact model. Otherwise, it is an exact representation.

Considering exactness, it should be mentioned that only the voxel-based representation is not exact because it uses discretization for classification of the voxels. Because surface-based and voxel-based models produce all the geometric information of the resulted 3D shape, they are complete, while the others are not.

However, it is not possible to select the one representation as the best model because each model has its own strengths and weaknesses. Deciding about the best model only depends on the intended application.

Table 2-1. The comparison of different visual hull models.

Model	Voxel-Based	Surface-Based (Polyhedral)	Image-Based	Bounding Edge
Geometric Entity	Cube Cells	Polyhedron Faces, Edges and Vertices	Two Dimensional interval samples	Bounding Edges
Number of Dimension	3D	2D	1D	1D
Exactness	Inexact	Exact	Exact	Exact
Completeness	Complete	Complete	Incomplete	Incomplete
Computational Complexity	Low	High	Low	Moderate
Storage Requirement	Moderate-High	High	Low	Low-Moderate

2.7. Stereo Vision

To reconstruct a 3D shape of the object, the second suggested model is proposed based on *Depth Maps*. Depth map is a map which contains the depth information for each pixel. Depth value is the amount of shifting between the positions of the corresponding pixels between two images from different views. The shifting amount is called *disparity*, and the depth map is also called *disparity map*. Depth maps can be shown as grayscale images, in which the nearer objects to the camera look lighter. The top performers of the existing methods in 3D object reconstruction are depth map based methods, which usually have two steps, producing depth maps and merging them.

The main step of depth map based methods is to produce the reliable depth maps, which influences the quality of the final reconstructed object. To produce the depth maps, two images of the scene from different viewpoints are used. Because of using a pair of images, it is called *stereo vision* or *stereopsis*. Stereo vision gets two rectified images from different viewpoints, and calculates the disparity for each pixel. Disparity value is the shifted amount between the two views for each pixel. In other words, disparity value is the difference between the position of each pixel in one view and its best match in another view. Because to provide the depth maps, the disparity value is calculated for all pixels of the image, it is called *dense stereo matching*.

In stereo matching, the view for which the depth information is calculated is called *reference view*, and another view is called the *target view*. To find the best match for each pixel of the reference view, first a neighborhood window is considered, which is usually a square window. Then a measurement is applied to find the best match pixel from the target view. There are two types of measurement including the error measurement and correlation measurement.

1. Error Measurement: It measures the errors between the reference pixel and the target one over the neighboring window. Finally, the target pixel which has the minimum amount of error is selected as the best match. *Sum*

of Absolute Differences (SAD) and *Sum of Square Differences (SSD)* are the error measurement functions which have been shown as follows.

$$SAD(x,y,d) = \frac{\sum_{i=-\frac{m-1}{2}}^{\frac{m-1}{2}} \sum_{j=-\frac{m-1}{2}}^{\frac{m-1}{2}} |R(x+i,y+j) - C(x+i+s*d,y+j)|}{m \times m} \quad (1)$$

$$SSD(x,y,d) = \frac{\sum_{i=-\frac{m-1}{2}}^{\frac{m-1}{2}} \sum_{j=-\frac{m-1}{2}}^{\frac{m-1}{2}} \left(R(x+i,y+j) - C(x+i+s*d,y+j)\right)^2}{m \times m} \quad (2)$$

where the neighboring window is $m \times m$ square window, and $R(x,y)$ and $C(x,y)$ mean the pixel value of the reference image and target image for row x and column y, correspondingly.

2. Correlation Measurement: It measures the correlation between two windows, and selects the pixel with the highest correlation value as the best match. Normalized Cross Correlation (NCC) is a correlation equation which has been shown as follows.

$$NCC(x,y,d) = \frac{\sum_{i=-\frac{m-1}{2}}^{\frac{m-1}{2}} \sum_{j=-\frac{m-1}{2}}^{\frac{m-1}{2}} (R(x+i,y+j) - \bar{R}) \times (C(x+i+s*d,y+j) - \bar{C})}{\sqrt{\sum_{i=-\frac{m-1}{2}}^{\frac{m-1}{2}} \sum_{j=-\frac{m-1}{2}}^{\frac{m-1}{2}} (R(x+i,y+j) - \bar{R})^2 \times \sum_{i=-\frac{m-1}{2}}^{\frac{m-1}{2}} \sum_{j=-\frac{m-1}{2}}^{\frac{m-1}{2}} (C(x+i+s*d,y+j) - \bar{C})^2}} \quad (3)$$

where $\bar{R}$ and $\bar{C}$ are the average of the pixel values over the neighboring window of the pixels of the reference image and target image, correspondingly.

Sample image pairs of Middlebury stereo vision datasets [24] have been shown in Figure 2-20. As it can be seen clearly, the objects which are nearer to the cameras have larger amount of shifting between two views.

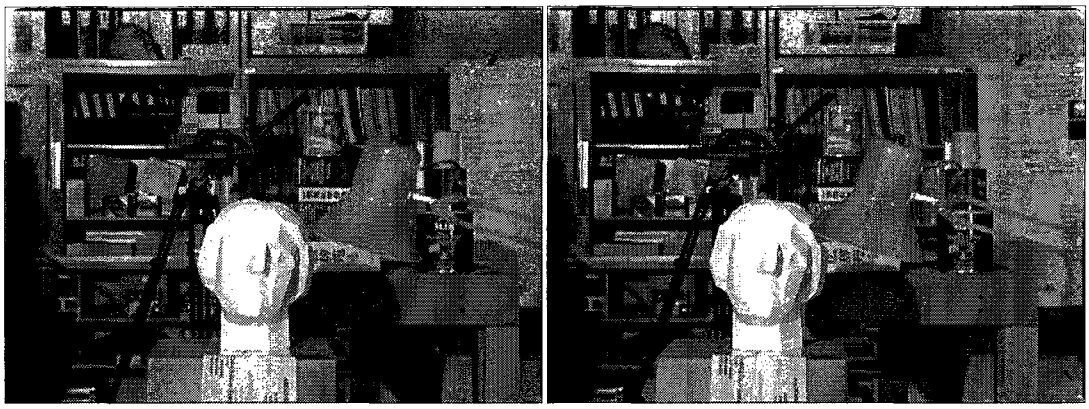


Figure 2-20. Sample image pairs for stereo vision [24].

The ideal depth map for the sample image pairs which have been shown in Figure 2-20 has been shown in Figure 2-21. Because the disparity values for the nearer objects are higher, they look lighter in depth map.

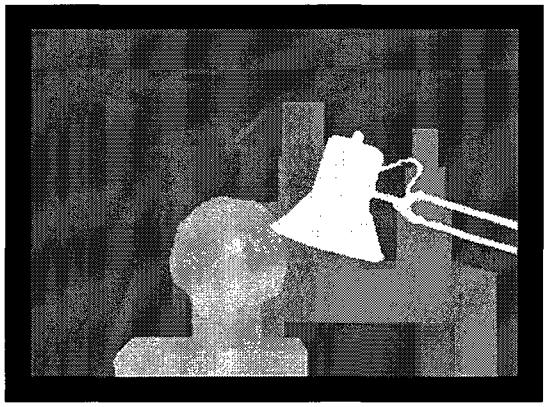


Figure 2-21. The ideal depth map for sample pairs [24].

The depth map based models have been reviewed in depth because the proposed model is a depth map based model. The proposed model uses the surface reconstruction methods to provide a triangular mesh surface to evaluate the results by Middlebury benchmark.

3. Implementation Platform

In this thesis, two models to reconstruct the 3D object have been proposed. The proposed algorithms are implemented using Java programming language and Matlab. Matlab is used because it is very fast for matrix manipulation which is very important in computer vision; images are considered as matrices. However, Matlab is very slow for other computations such as ray projection. In these computations, the algorithms are implemented in Java.

To test and evaluate the proposed models, the algorithms are applied on existing datasets, which are popular in this field. The selected datasets will be described in the following subsections. For one of the datasets, Adobe Photoshop is used to produce the high quality silhouette images as a semi-automatic process, like what S. Lazebnik et al. [25] did.

3.1. Implementation Methodology

As it mentioned before, matrix manipulations is implemented in Matlab, which include the silhouette production, image rectification, and stereo matching process. The codes for mentioned computation are implemented as the Matlab functions. For each dataset, a final script is implemented which performs all the steps for all the images sequentially. However the number of implemented functions in Matlab is great, all the implemented functions in Matlab have small number of *Lines of Code (LOC).*

Source line of code (LOC) is a software metric which is used to measure the size of an implemented application, by counting the number of lines in the source code of the implementation. There are two types of this measure, physical and logical. Physical line of code which is referred by LOC counts the number of line in the text file of the source code. It counts the comment lines and also blank lines as lines of code, which is not accurate enough to estimate application size. In contrast, logical line of code, referred as

LLOC, counts the number of statements in the code. Logical measure is more appropriate for size estimation of the application.

However, all the other computations of the proposed methods are implemented in Java using *Object Oriented Programming (OOP)*. I defined 31 different classes which are described in Appendix A. Just to show the estimation of the implemented codes, it should be mentioned that the number of logical lines of code (LLOC) for all the java codes is 4252 lines, and the number of physical lines of code (LOC) is 6380. For detailed information of the implemented classes, please refer to Appendix A.

Another important issue here is that there is no graphical user interface implemented in Java, and codes are just implemented to calculate the final results and save it as a file with PLY format. Finally, MeshLab software [26] is used to show the final result. MeshLab is an advanced mesh processing application for automatic and user assisted editing, painting, converting, cleaning, remeshing, coloring, filtering, measuring, scanning, and rendering of large unstructured 3D triangular meshes.

Implementation of MeshLab is started as a university project with small group of core developers at Visual Computing Lab of the Italian National Research Council Institute. Now, there are many plug-in developers for MeshLab around the world. Figure 3-1 shows a sample view of MeshLab application showing the final result of the proposed visual hull model on Dinosaur dataset.

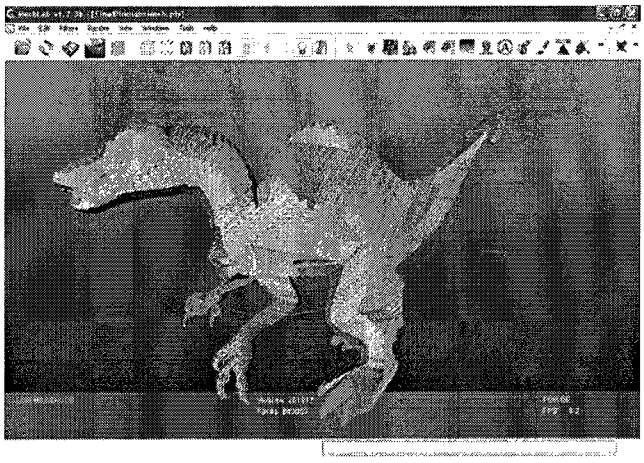


Figure 3-1. A sample view of MeshLab application.

PLY file format is developed at Stanford University [27]. PLY file format has two different types, binary and ASCII. An ASCII PLY file is a text file which first determines the number of vertices and surfaces of a mesh, following by the information of all vertices and surfaces. Vertices are defined by their x, y, and z parameters and their color if applicable. Surfaces are determined by the lists of their vertices which are defined by their indices in vertices section.

3.2. Datasets

To show the performance of the proposed models, complex 3D datasets have been selected. These datasets are the 3D Photography datasets [7] and Middlebury datasets [2]. 3D Photography datasets are produced using three fixed cameras (Canon EOS 1D Mark II) and a motorized turn table in Beckman Institute and Department of Computer Science at University of Illinois at Urbana-Champaign. Each dataset of 3D Photography collection has 24 images from 24 points of view, which are calibrated using Intel's OpenCV package [28]. The calibration information is provided in the format of the Camera Calibration Toolbox for Matlab [29]. Moreover, the contour information of the interesting object has been provided as unconnected 2D pixels.

Middlebury datasets are provided by support of Middlebury College, Microsoft Research, and the National Science Foundation. They used the Stanford Spherical Gantry to capture images, which enables moving a camera on a sphere to specified latitude and longitude angles. The cameras are calibrated by capturing the images of a planar grid from different points of view.

From 3D Photography datasets, Dinosaur dataset and Predator dataset are selected, each of which has 24 images from different viewpoint. The intrinsic parameters and the image size are the same for the first 8 views. They are the same for next 8 ones as well as the last 8 ones. These intrinsic parameters include the focal length, principal point, skew coefficient and distortion coefficients. It is easy to provide the matrix of the intrinsic parameters to map the camera coordinate system to the pixel coordinates of the image. The extrinsic parameter for each camera is provided as a 3×4 matrix. This matrix

can be used to map a world point in homogeneous coordinate to the corresponding camera coordinate. The images for Dinosaur dataset have been shown in Figure 3-2.

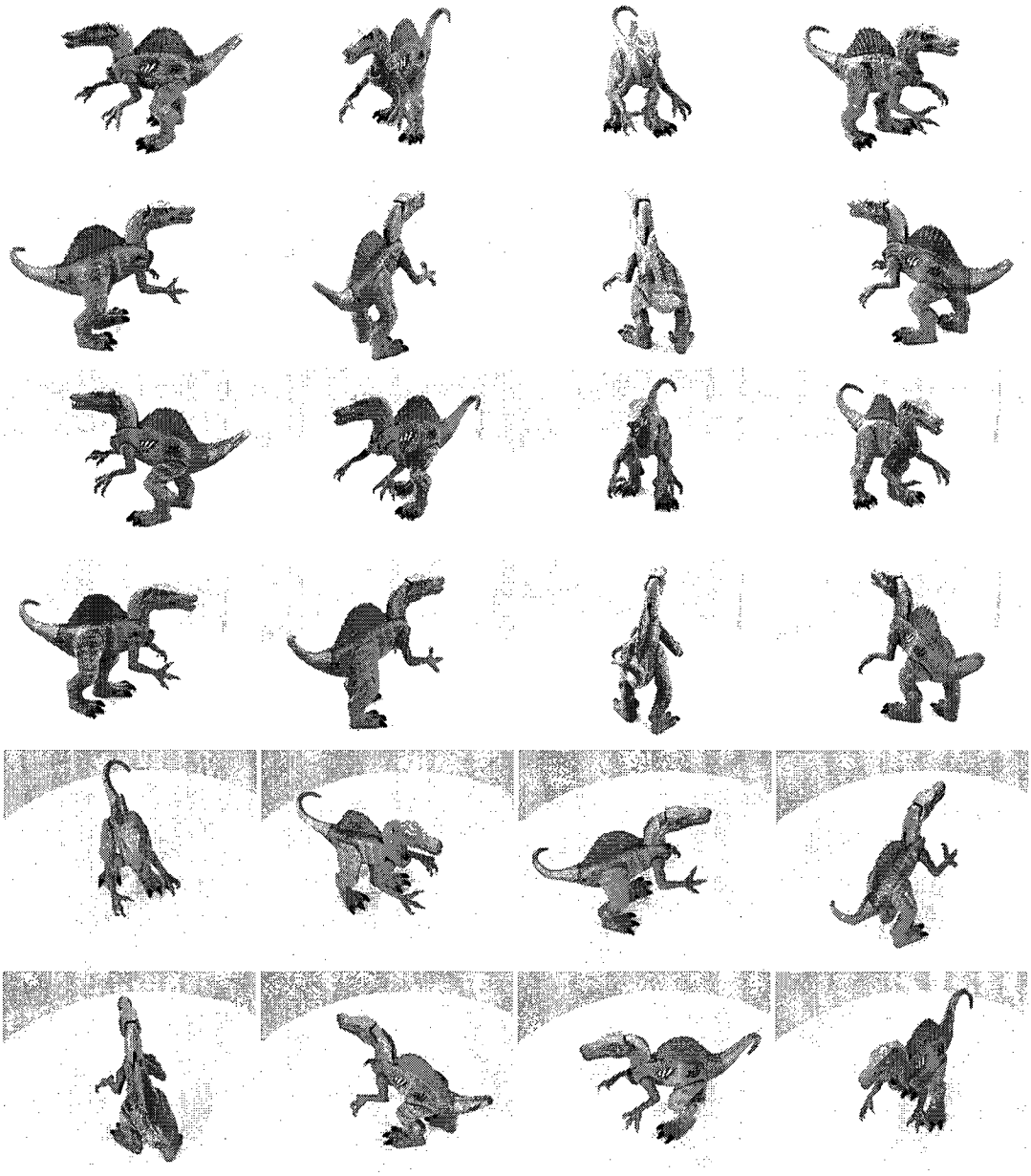


Figure 3-2. Images of Dinosaur dataset.

The Predator dataset specification is similar to Dinosaur dataset. The images for Predator dataset have been shown in Figure 3-3.

Figure 3-3. Images of Predator dataset.

From Middlebury datasets, the DinoSparseRing dataset is selected which has 16 images from different viewpoint. The intrinsic parameters and image size is the same for all 16 views. Figure 3-4 shows the images for DinoSparseRing dataset.

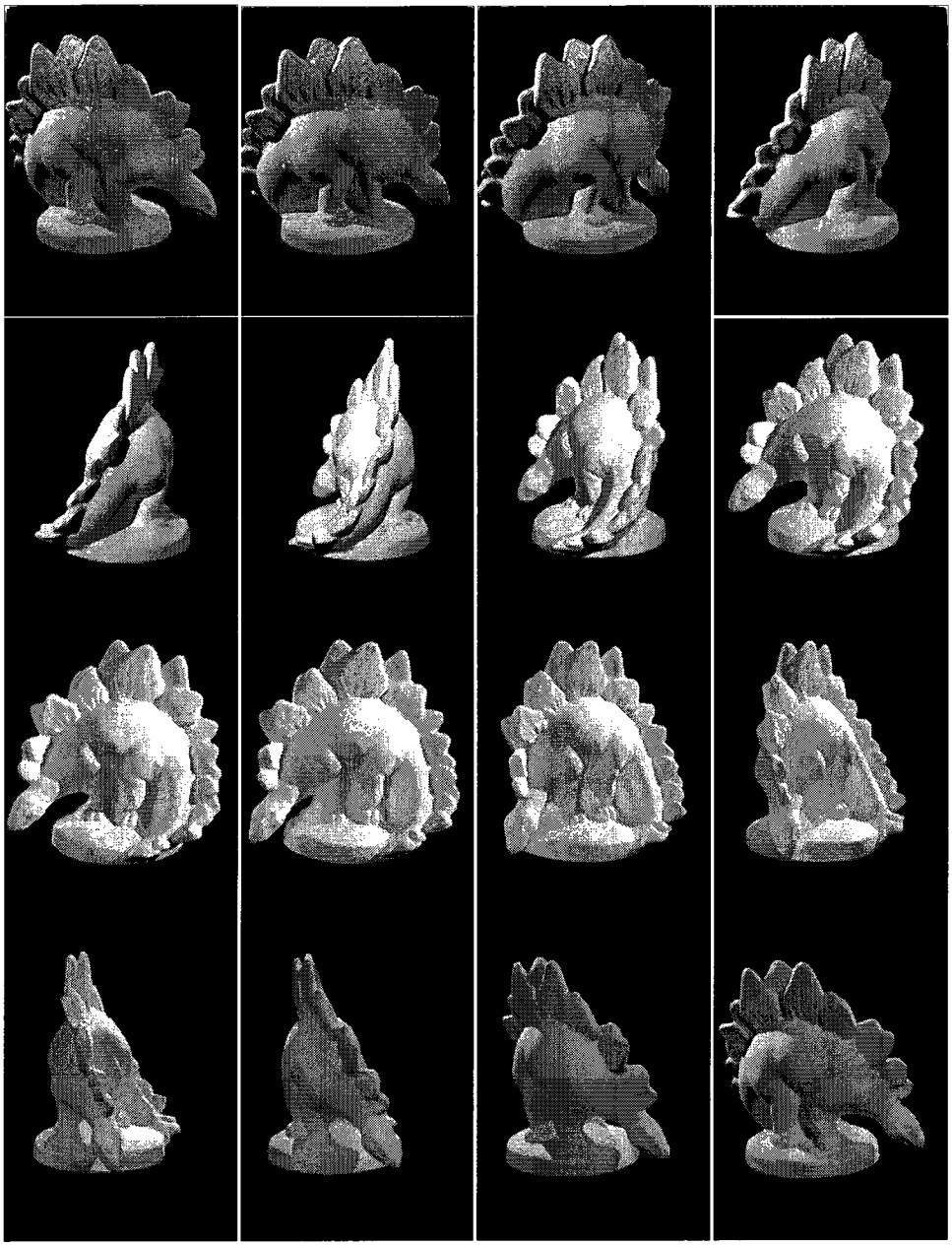


Figure 3-4. DinoSparseRing dataset images.

The contour information is provided for each image of 3D Photography dataset in a text file in the following structure. The file starts with "Contour" name, followed by the number of contours. It contains the number of pixels and the pixel information for each contour. In all of the datasets, it is considered that there is only one contour. As it can be seen in some images of Dinosaur dataset, the images have some holes, so the silhouette

information of this object should not be a connected part. However, they consider the contour of the object as a part without any hole.

A sample file is as follows.

```
1.    CONTOUR
2.
3.    1
4.
5.    951
6.
7.    1276.27 871.568
8.    1280.03 871.535
9.    1283.78 871.377
```

As it is clear, the pixels coordinate information is not discrete, it is in float format. Prior to computing a connected version of the image contour, the numbers should be discretized.

3.3. Silhouette Images Computation

This section is divided into two parts. The first one is for the 3D Photography datasets, and the second one is for the DinoSparseRing dataset.

For 3D Photography datasets, the contour information is provided. I use this information to compute the silhouettes. I first convert the pixel information of the contour to the discrete values. Using these discrete values, I produce a binary image which is black in the mentioned pixels, while other pixels are white. I connect each pixel to the consecutive pixel by finding the best discrete connection throw four neighboring pixels. This algorithm works based on the slope of the connecting line between the current pixel and the consecutive one. In each iteration, it selects the best of its four neighbors. The best neighbor is one for which the slope of the connecting line throw the current pixel is close to the slope of the line connecting the current pixel and the consecutive one.

After producing the closed silhouette contour, the silhouette image can be produced. In other words, the pixels which are located in the contour should be

considered as foreground pixels. To do so, I implement another algorithm which classifies pixels based on the class of its neighbors. If one of the eight neighboring pixels of a pixel is classified as foreground, the pixel will be classified as foreground too. This is the same for the background pixels. If all the neighboring pixels are not classified yet or they are the contour pixels, the pixel will be classified based on the class of the pixels in the other side of the contour pixels, inversely.

These two mentioned algorithm is implemented in Matlab as a parser function which gets the text file as an input and produces the silhouette images. Resulted silhouette images for Dinosaur dataset have been shown in Figure 3-5, and Figure 3-6 shows the produced silhouette images for Predator dataset.

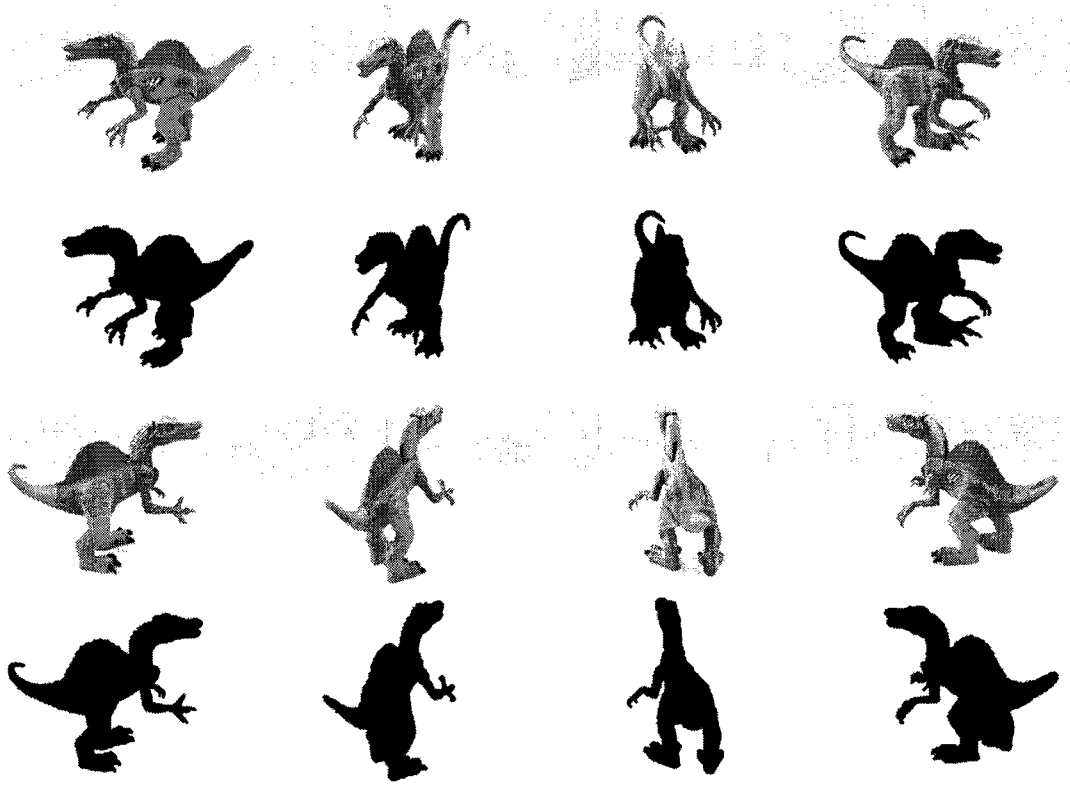


Figure 3-5. Resulted silhouette images for Dinosaur dataset.

As it can be seen clearly, silhouette images do not have any holes, because there is no information provided for the exiting holes. To apply the proposed models, the same silhouette images as ones shown in Figure 3-5 and Figure 3-6 are used.

For Middlebury dataset, there is no contour information, but dataset providers suggest doing three steps to get a good set of silhouettes. However, the results of their suggestion are not good enough to get appropriate results from the proposed models. I refine these results using Adobe Photoshop as a semi-automatic process to get high quality silhouette images. Using the new silhouette images, many of the wrong holes produced by the previous silhouette images are removed from the 3D reconstructed object. Like how S. Lazebnik [25] used Adobe Photoshop, I use it as a segmentation of the foreground and background pixels.

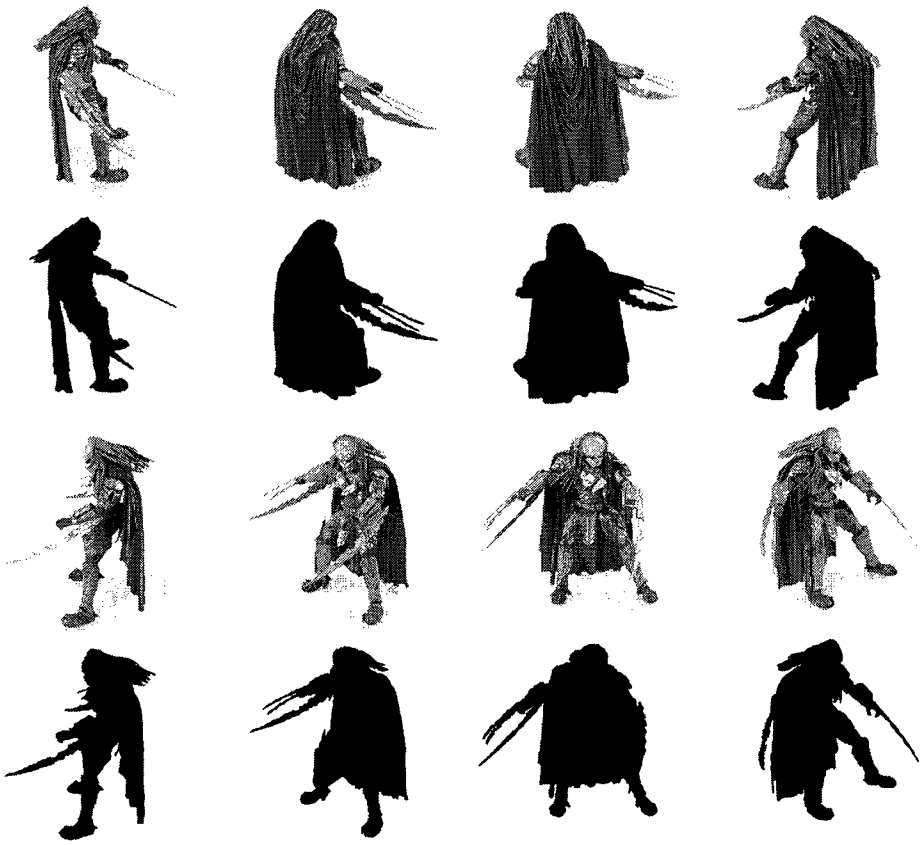


Figure 3-6. Resulted silhouette images for Predator dataset.

Figure 3-7 shows the results of the suggested method as well as the refined results. The silhouettes resulted of the suggested method has many errors, especially for dark shadows on the object. However, the object is partially outside the field of view in some images, which makes some inconsistency to the final results.

The suggestion of the providers of Middlebury dataset [2] includes following steps:

1. Threshold the gray images at 0.19 (where intensity values range from 0 to 1).
2. Dilate the result by 10 pixels.
3. Erode the result of dilation by 7 pixels.

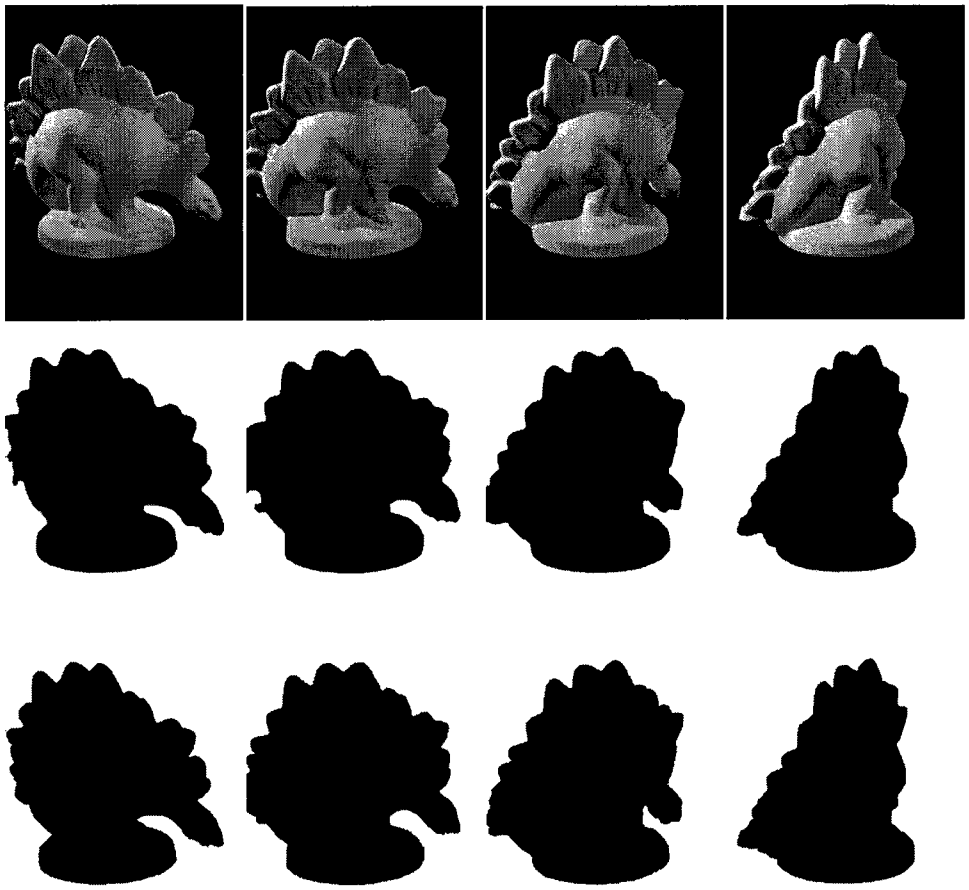


Figure 3-7. Resulted silhouette images for DinoSparseRing dataset; sample images (1st row), results of providers suggestion (2nd row), manually refined results (3rd row).

4. Proposed Visual Hull Model

In this section, 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.

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 study. 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.

The proposed visual hull model is described in the following subsection in details. The resulted visual hull model has been shown in next subsection, followed by the evaluations and comparison of the results by modern approaches.

4.1. The Algorithm

As it can be seen clearly in the comparison of existing models in previous sections, every visual hull model has 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 in 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 by Projective Visual Hulls which is published by Lazebnik et al. [25]. 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 produces 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 outputs of the first steps for both models 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 which is refining the final model is the same for both models.

4.1.1. Modified Bounding Edge Model

The first step of the algorithm is to apply the bounding edge model to the silhouette set. 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 between these two types 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. [22] 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 $\Pi_k(P)$.

$$R_i^k = \{P | \Pi_k(P) = p_i^k\} \quad (4)$$

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. These returned intersection parts 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 a 1D value (real number). Bounding edge E_i^k is shown by

$$E_i^k = \left\{ \left(SP_{i,m}^k, FP_{i,m}^k \right) \middle| m = \{1, \dots, M\} \right\} \quad (5)$$

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 a 1D value, but in the modified model, each endpoint needs to have sufficient space for the 1D value, the silhouette reference, and the pixel position. Like the main bounding edge model, the modified bounding edge information should be produced based on each point of view.

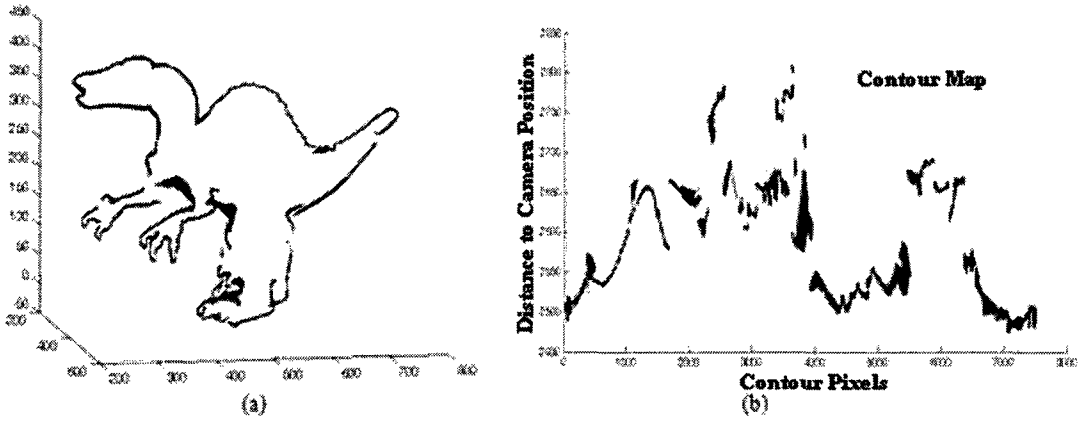


Figure 4-1. (a) The bounding edge model for the 1st view of Dinosaur dataset and (b) corresponding contour map.

Figure 4-1 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. [11] 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

$$\exists O \; \forall k \in \{1, \ldots, K\} \quad \Pi_k(O) = S_k. \qquad (6)$$

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.

Table 4-1 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.

Table 4-1. 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

Figure 4-2 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.

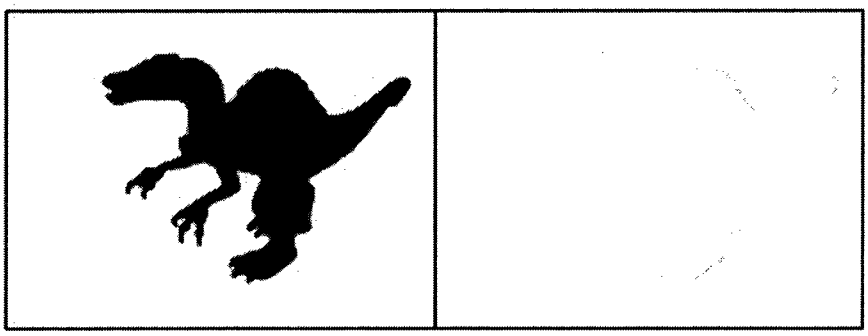


Figure 4-2. The silhouette of the 1st view of Dinosaur dataset (left) and the inconsistent pixels which are the difference between the input silhouette and the consistent one (right).

The second type of discontinuity is due 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 hands 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 hands. These discontinuities can be seen in the contour map as well. Actually, these discontinuities are due to the fact that the occluded parts of this view are visible from other points of view. So the discontinuities are recovered for the final visual hull by the occupancy intervals from the other points of view. It should be mentioned here that the occluded parts of the 3D object which are not visible in all views do not make any discontinuity in contour map.

4.1.2. 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 for each point of view. These surfaces are bounding surfaces which cover parts of the object which are invisible for the corresponding point of view. The input for this step is a contour map, and the output is 3D triangular mesh surfaces. The algorithm for this step 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 1D value (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 $(SP_{i,m}^k, FP_{i,m}^k, FP_{i+1,n}^k)$ and $(SP_{i,m}^k, FP_{i+1,n}^k, SP_{i+1,n}^k)$. 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.

4.1.3. 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.

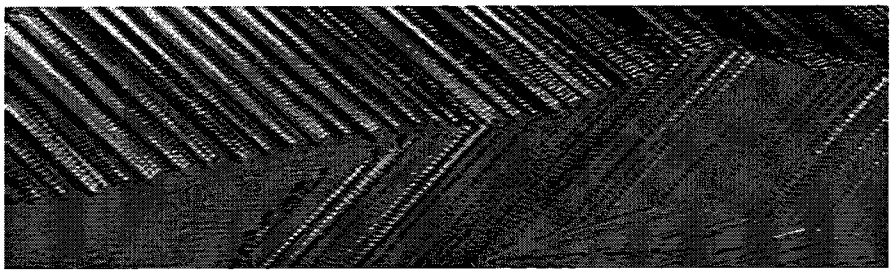


Figure 4-3. Intersection of the occupancy intervals form different viewpoints.

Figure 4-3 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. At first glance, 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. Since 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.

4.1.4. 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, collapse and swap operations are applied on the final mesh. Edge split operation is considered for too long edges, while edge collapse operation is performed for too short edges. The edge swap operation guarantees that each vertex has a degree close to six. After applying the re-meshing step, we will have a refined complete triangular mesh of the visual hull.

4.2. Experiments

To show the quality of the proposed model, it is applied to some datasets which are describe in previous section. The results have been shown in the next subsection followed by an evaluation part. There is one step before applying the proposed model which is producing consistent silhouette set for each dataset based on provided information, which is described completely in previous section.

4.2.1. Results

Figure 4-4 shows the bounding mesh surfaces resulted from the first 8 views of Dinosaur dataset. 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.

Figure 4-4. Bounding surfaces resulted for the first 8 views of Dinosaur dataset.

The bounding surfaces resulted for the Predator dataset has been shown in Figure 4-5, and Figure 4-6 shows the same type of results for DinoSparseRing dataset.

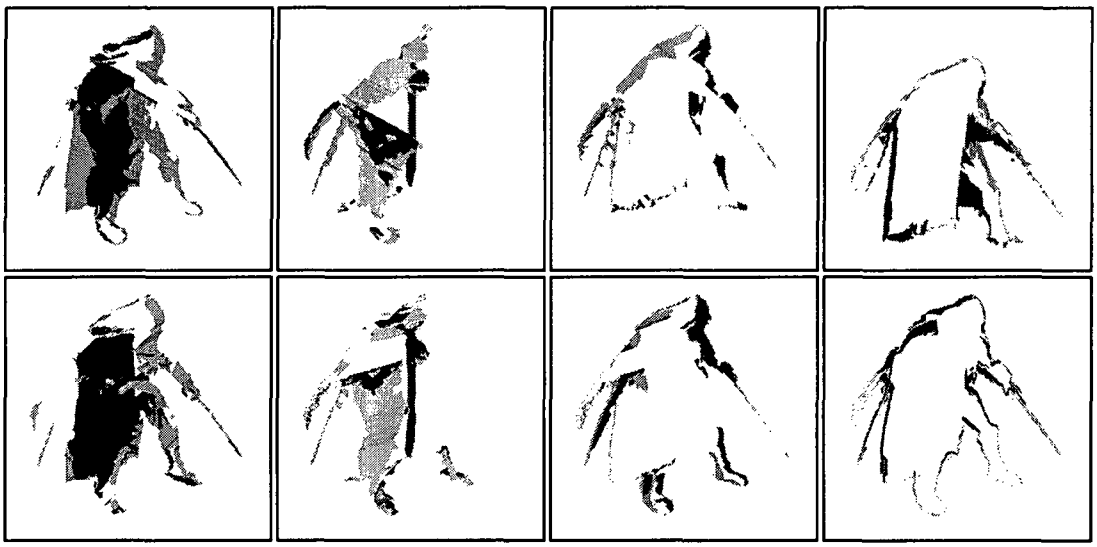


Figure 4-5. Bounding surfaces resulted for the first 8 views of Predator dataset.

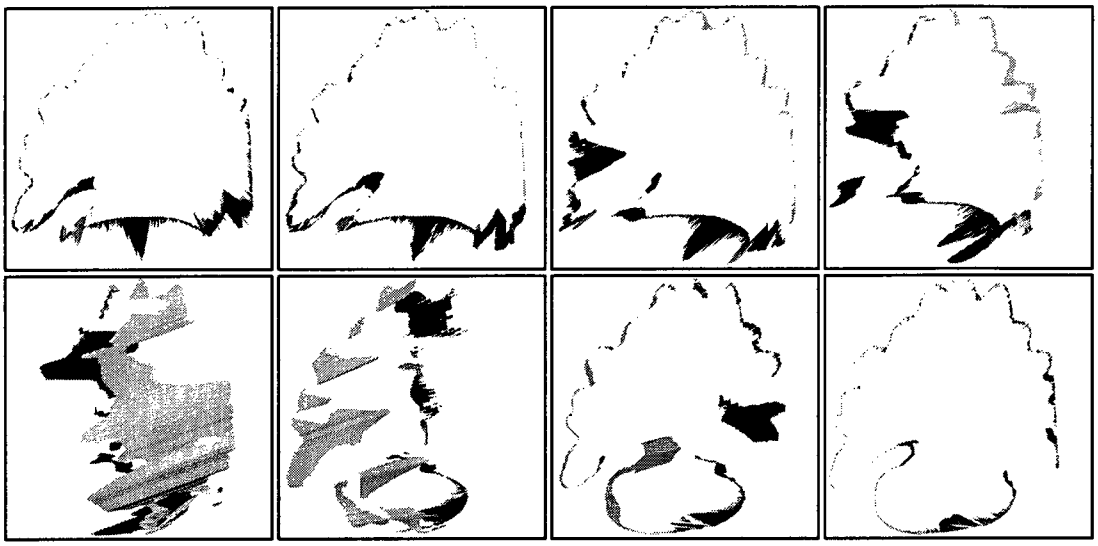


Figure 4-6. Bounding surfaces resulted for the first 8 views of DinSparseRing dataset.

Figure 4-7 and Figure 4-8 show the merged surface of the bounding surfaces for Dinosaur and Predator datasets. The final triangular mesh for DinoSparseRing dataset has been shown in Figure 4-9. As it can be seen clearly, the surfaces are connected and the discontinuities have been removed from the mesh.

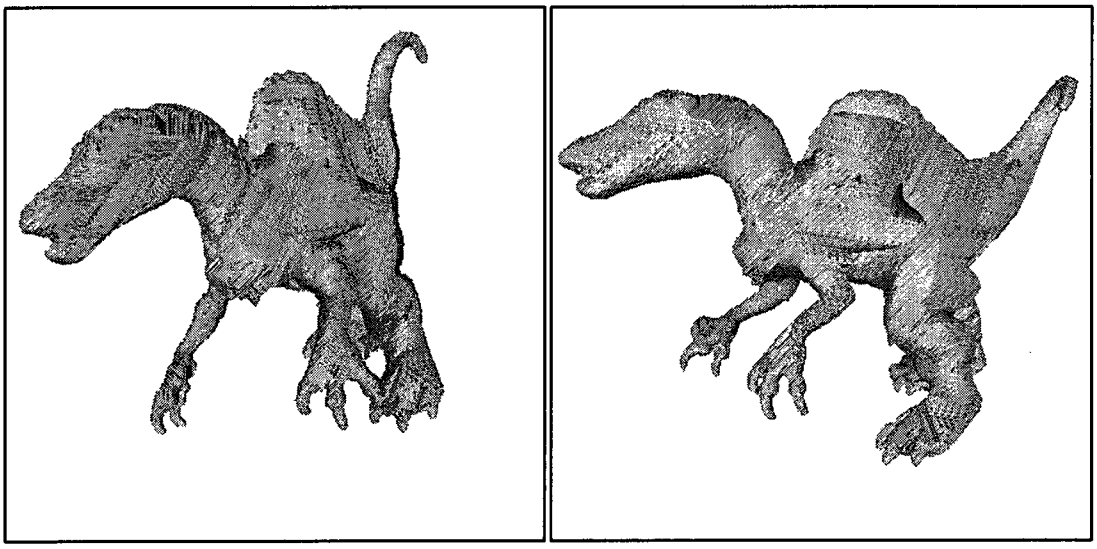


Figure 4-7. Final triangular meshes for Dinosaur datasets.

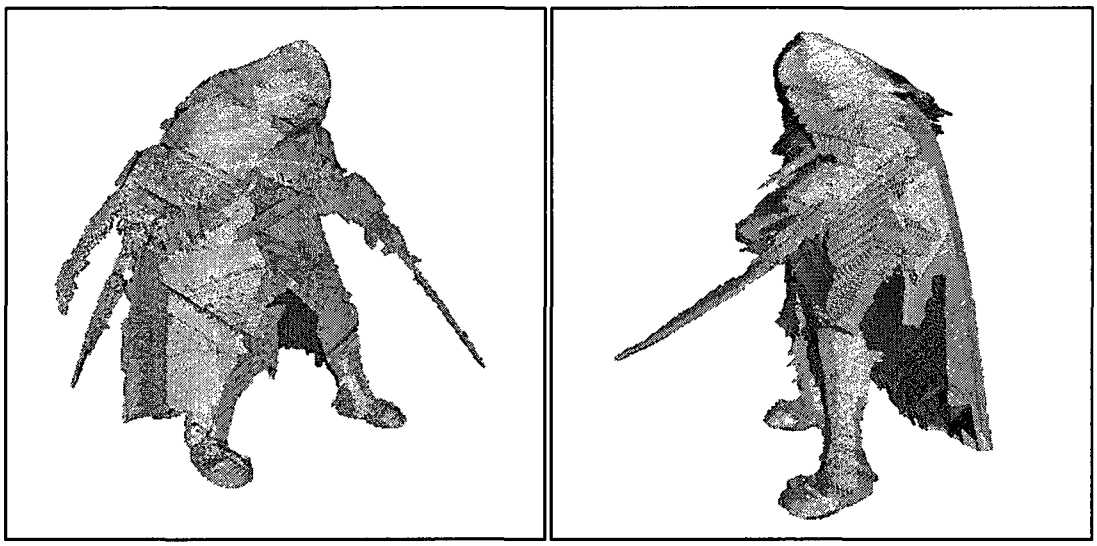


Figure 4-8. Final triangular meshes for Predator datasets.

Number of vertices in the overall surface and merged surface before re-meshing for each dataset has been shown in Table 4-2. By merging surfaces, number of vertices is decreased significantly. For example, for Dinosaur dataset, it has been decreased by 40%. It is true that some points in the final mesh are removed because they are identical in two or more viewpoints.

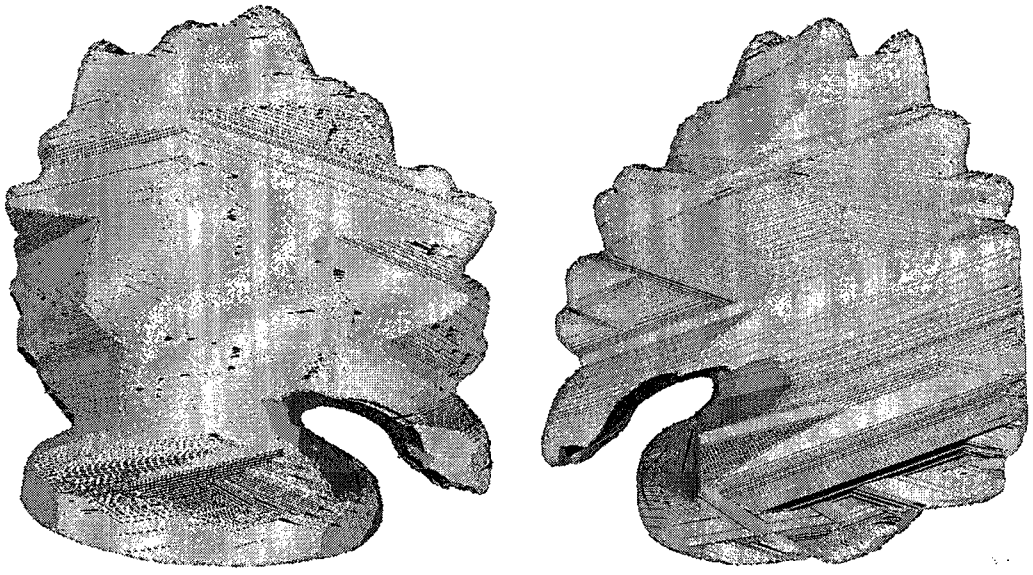


Figure 4-9. Final triangular meshes for DinoSparseRing dataset.

Table 4-2. 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

4.2.2. 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 study, the projective visual hull model and the last two versions of Exact Polyhedral Visual Hulls [13] are selected for comparison. The results for other models are taken from Lazebnik et al. [25] 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 4-3. 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 4-3. 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 [25]. 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 4-4. The big 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 4-4.

Table 4-4. 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

4.3. Conclusion

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 available models. The execution time of the proposed model is close to the time required for bounding edge model. Although the storage requirements are 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.

5. The Proposed 3D Object Reconstruction Model

The goal of the 3D object reconstruction models is to reconstruct a 3D shape of the object using multi-view calibrated images. In recent years, many high quality models have been proposed that are more sophisticated than the early algorithms. Early algorithms match and reconstruct the 3D points of the object surface independently, while recent methods define the problem as a global energy minimization function, which leads to a better quality and performance. Because the existing algorithms in this field use stereo vision to find the depth value for each pixel, this field is also called *Multi-View Stereo (MVS)*.

5.1. Existing Models

The existing methods are surveyed by S. M. Seitz et al. [30]. In this survey, six fundamental properties are defined to categorize the existing approaches which are as follows:

1. Scene Representation: The 3D reconstructed object can be represented in many geometrical ways such as voxel representation, triangular meshes, and depth maps.
2. Photo-Consistency Measure: The reconstructed object should be compatible with the input images. Existing approaches evaluate the compatibility by different measures which are called photo-consistency measures. These measures include the correlation measures used for comparing pixels of different images. For example, *Sum of Squared Differences (SSD)* and *Normalized Cross Correlation (NCC)* are photo-consistency measures.
3. Visibility Model: The visibility issue is very important in multi-view framework, since to use the photo-consistency measures, only those views that the 3D point is visible for them should be considered.

4. Shape Prior: Some existing approaches uses shape priors to reconstruct the 3D model with some appropriate specification.
5. Reconstruction Algorithm: This property is very important, which is the base of each model. The existing algorithms are divided into four categories including
    a. 3D Volumetric Approaches
    b. Surface Evolution Techniques
    c. Feature Extraction and Expansion Algorithms
    d. Depth Map based Methods

    These categories are described in details in following subsections.
6. Initialization Algorithms: Some models need more information of the object. For example, many algorithms need only a bounding box or volume of the object. Some algorithms use the silhouette information in their algorithm, so they require the high quality silhouette images.

Moreover, S. M. Seitz et al. [30, 2] provided benchmark datasets to evaluate and compare the existing models. For each dataset, the ground truth 3D mesh model is provided which was capture using a Cyber-ware Model 15 laser strip scanner by a resolution of *0.25mm* and an accuracy of *0.05-0.2mm*. Based on the provided ground truth, the results of all models can be evaluated and compared with each other. They get the result of any model, compare it with the ground truth, evaluate their measures, and upload to a website, which is provided to compare the existing models. The most important issue here is that they only accept the results as a 3D triangular mesh. The ground truth for the DinoSparseRing dataset has been shown in Figure 5-1.

S. M. Seitz et al. [30] defined two measures to evaluate the results quantitatively, *accuracy* and *completeness*. The definition of these measures is as follows.

1. Accuracy: Like the other accuracy measures, it shows the difference between the real object which is the ground truth and the calculated result which is reconstructed object. In other words, it determines how close the reconstructed object is to the ground truth.

To compute the accuracy measure, first the distance between the points of the reconstructed object and the nearest points of the ground truth is calculated. Then a statistical summary of the distances is provided, which computes distance d such that X% of the points of the reconstructed object are within distance d of points of ground truth. For example, we can use 90% as variable X.

Figure 5-1. Ground truth for DinoSparseRing dataset [2].

2. Completeness: This measure determines how much of the ground truth is reconstructed by the model. In this case, the distance from the ground truth to the reconstructed object is calculated, which is opposite of measuring accuracy.

   Now, the statistical summary of the distances computes the fraction X of the points of the ground truth which are with in distance d of the points of reconstructed model.

Before describing the proposed model, the existing approaches are reviewed in the following subsections, and the next subsections show the results and the evaluation of the proposed model. Existing approaches are divided to four categories which are described in details.

5.1.1. 3D Volumetric Approaches

3D volumetric approaches first define a cost function over a 3D volume, followed by the surface extraction. Voxel coloring algorithm is a sample of these approaches which is introduced by S. Seitz and C. Dyer [31]. In voxel coloring, first scene is discretized into a set of voxels which are traversed and colored in depth order. The problem is to assign colors to the voxels in a 3D volume to maximize photo integrity with the input calibrated images.

P. Song et al. [32] proposed a 3D volumetric method, which is not published yet, but the results are available in Middlebury benchmark.

5.1.2. Surface Evolution Techniques

The second category of the 3D object reconstruction is surface evolution techniques, which iteratively evolve a surface to minimize the cost function. This category can be divided into three classes itself, based on their geometric entity including voxels, level sets, and surface meshes. Space carving methods, which are voxel-based, consider an initial volume, and try to carve the inconsistent voxels. Level set methods define a set of partial deferential equations on a volume, and by shrinking and expanding the volume; they try to minimize the equations. The last class works on the evolving mesh by defining the internal and external forces.

A. Auclair et al. [33] proposed a surface evolution method which drives the deformation of a mesh towards using Scale Invariant Features Transform (SIFT) descriptor. The proposed method uses SSD to recover the small scale details. Y. Furukawa and J. Ponce [34] used rims over the surface of the object to propose a surface evolution approach. They first initialized the 3D shape of the object by its visual hull. Then, the resulted model is carved by maximizing a photometric consistency score. C. Hernandez and F. Schmitt [35] proposed an algorithm to reconstruct the 3D geometry as well as the texture. To evolve the surface, two external forces are defined which are a texture driven force and a silhouette driven force.

A. Zaharescu at al. [36] proposed *TransforMesh* which is a mesh based surface evolution method capable of handling topology changes in evolution as well as removing the self intersection of the reconstruction.

K. Kolev et al. [37] proposed a surface evolution method which uses a continuous global optimization of an energy function. A. Ladikos et al. [38] proposed a graph cut method which avoids the local minima in narrow band around the current surface estimate. Method proposed by J.-P. Pons et al. [39] minimizes the prediction error of the shape and motion estimates. Other graph cut approaches are proposed by S. Tran and L. Davis et al. [40] and G. Vogiatzis et al. [41].

Two other surface evolution approaches [42, 43] are proposed, the results of which are available in Middlebury benchmark without the name of the authors.

5.1.3. Feature Extraction and Expansion Algorithms

In this category, first a set of features is extracted based on the input images. After feature matching among the images, features are reconstructed, which is followed by providing a surface to fit the reconstructed features.

Y. Furukawa and J. Ponce [44, 45] proposed a 3D object reconstruction model using feature extraction, expansion and filtering. First, a sparse set of patches are produced using matching the extracted features found by Harris feature extraction [46] and Difference-of-Gaussians operator. Then using expansion, the initial matches are spread to the nearby pixels, followed by a filtering step which removes the incorrect matches.

M. Jancosek et al. [47] proposed a scalable method which is able to produce the 3D reconstruction using large amount of data. The result of the algorithm is obtained in acceptable time and required accuracy. However, it is not the optimal result in case of accuracy. The basis of the algorithm is like other methods in this category. It finds the matches between the extracted features of different images, and produces the 3D

geometry of each match which is called 3D seed. The expansion of the 3D seeds is called growing which is followed by the filtering step.

J. Starck and A. Hilton [48] proposed a *surface capture system* which produces animated content automatically from multiple video cameras from different viewpoints. For each time instance, the visual hull of the object is created, followed by matching extracted feature from different viewpoints. Then a surface reconstruction method produces 3D reconstructed shape of the object. Finally, merging 3D reconstructed shapes provides a 3D video representation which is also called free-viewpoint video. In free-viewpoint videos, users have the control over the camera viewpoint.

A. Delaunoy et al. [49] and P. Gargallo et al. [50] proposed a surface evolution model which minimizes the reprojection error. Reprojection error is the difference between the input images and the images produced by projecting the reconstructed 3D object into the all image planes from different viewpoints. Another surface evolution method is proposed by C. Strecha et al. [51] which models visibility and depth issues as a hidden Markov Random Field jointly.

5.1.4. Depth Map based Methods

The last category includes depth map based approaches. Depth map is a map which includes the depth information for each pixel. Depth maps can be shown as a grayscale image, such that the nearer objects to the camera look lighter. Depth map based methods usually have two steps including producing depth maps from different viewpoint and merging them. Most of the top performer algorithms for Middlebury benchmark [2] are depth map based methods.

R. Szeliski [52] proposed a depth map based method especially to predict the appearance of a novel view of the scene, and reconstruct the occlusions by comparing the depth maps of different views. P. Gargallo and P. Sturm [53] proposed a Bayesian 3D modeling which is also a depth map based model and uses an energy minimization method. Occlusion and outliers are managed by defining hidden visibility variables.

Bradly et al. [54] proposed a method which produces 3D points using binocular stereo matching, followed by point filtering. Another depth map based method is proposed by Liu at al. [55] which uses the visual hull information, frontier points and implicit points to merge the depth maps.

The depth map based algorithm proposed by M. Goesele et al. [56] uses a window based voting approach to produce the depth maps and a volumetric approach to merge them. Instead of using the disparity values between the images, the depth variable is defined as the distance between the point and the camera position, and based on each depth value, the correlation measurement is done between neighboring camera views to find the best depth value for each pixel.

Y. Liu et al. [57] proposed a continuous depth estimation method, instead of a discrete counterpart. Moreover, the patch based NCC measurement is applied to find the best matches between different views. K Li et al. [58] and Deng et al. [59] proposed another depth map based method which is not published yet, but their results are available in Middlebury benchmark. There is the result of another depth map based method [60] on Middlebury benchmark without the name of the authors.

The results of most of the reviewed existing models on Middlebury benchmark is available online which are denoted by the last name of the main author. There is one more method on Middlebury benchmark which is denoted by NIPS_829 [61]. However there is no information about the proposed model.

5.2. Surface Reconstruction Methods

Some existing algorithms first provide the 3D points with normal direction information, called *oriented points*, over the surface of the object, and then at final step they provide a mesh surface over the existing points. The critical issues in surface reconstruction are mentioned as follows.

1. The points are not distributed uniformly.

2. The position and direction of the points are noisy.
3. No information is provided for some parts of the surface.

Surface reconstruction methods should accurately fit the input points as well as removing the outliers and filling the existing holes. Delaunay triangulations and Voronoi diagrams are two samples of surface reconstruction. A Delaunay triangulation is a triangulation for a set of n-dimensional points, such that no point in the set is inside the circum-hypersphere of any simplex in the triangulation [62].

R. Kolluri et al. [63] proposed the spectral surface reconstruction. In this approach, first a Delaunay tetrahedralization is performed, followed by the spectral graph partitioning which decides about the position of the tetrahedrons. H. Hoppe [64] proposed a local method for the nearby points to estimate the tangent planes. M. Kazhdan [65] proposed a method based on Poisson problem. Poisson equation is a partial differential equation which is used widely in many areas such as computer graphics, electrostatics, mechanical engineering and theoretical physics. Because Poisson surface reconstruction considers all the points at once, it is robust to noise and non-uniform point cloud. The implementation of Poisson Surface Reconstruction is freely available online [66].

B. Curless and M. Levoy [67] proposed a volumetric surface reconstruction method, which is called *Vrippack* [68]. Vrippack is originally implemented for range images, whose implementation is freely available online. M. Goesele et al. [56] used Vrippack for merging depth maps. They obtained a good accuracy for their method on Middlebury benchmark, while in case of completeness, their results are so worse.

I used Poisson surface reconstruction as the final step of the proposed model which is described completely in following subsection, followed by its results and evaluation.

5.3. The Proposed Algorithm

Like other depth map based approaches, my proposed algorithm has two main steps; producing depth maps and merging them. Each step includes three parts which are as follows.

1. Producing Depth Maps
    a. Image Rectification for each pair
    b. Stereo Matching
    c. Producing 3D Points
2. Merging 3D Points
    a. Refine the Position of the 3D Points
    b. Remove Inconsistent Points
    c. Providing a Surface over the Points based on the Normal Directions

Each step is described in details in the following subsections.

5.3.1. Producing Depth Maps

This step is done for each image pair independent to the other views. The results of this step are the 3D points calculated for each viewpoint, which will be combined in the next step. First of all, the nearest camera for each view is selected as the target view for stereo matching.

The stereo matching is also called correspondence problem between two views, which is a problem of finding a corresponding point displayed by one view in the image of the other view. In most camera configurations, finding correspondence pixels requires a search in two dimensions. However, if the two cameras are aligned to have a common image plane, the search is simplified to one dimension; a line that is parallel to the line between the cameras (the baseline). *Image rectification* determines a transformation of

each image plane such that pairs of conjugate epipolar line become collinear and parallel to one of the image axes, usually the horizontal one.

The two sample views of Dinosaur dataset are selected as an image pair for stereo matching which have been shown in Figure 5-2. Because the image planes of the cameras are not on the same plane, the epipolar line for each pixel of reference view in target view is not parallel to the horizontal axes of the image which leads to in false matching results.

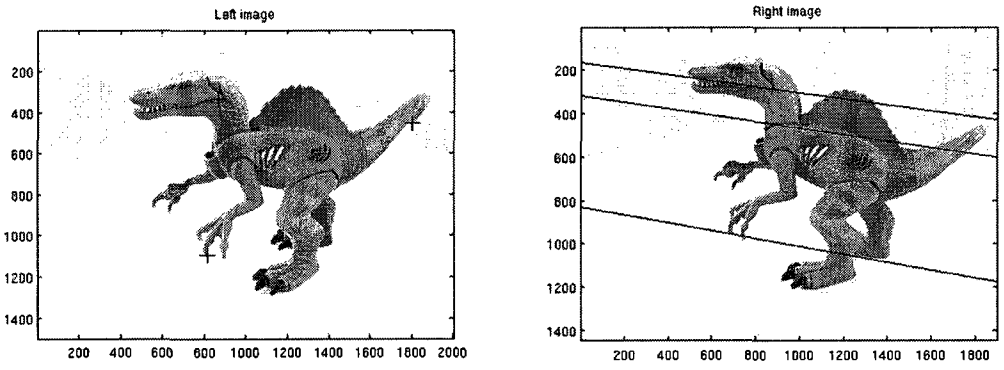


Figure 5-2. Dinosaur sample image pair.

So before starting the matching step, the images should be rectified. A. Fusiello et al. [69] proposed a simple method to rectify the image pairs. I implemented their method in Matlab, the results of which have been shown in Figure 5-3. The epipolar lines are parallel to the horizontal axes of the images.

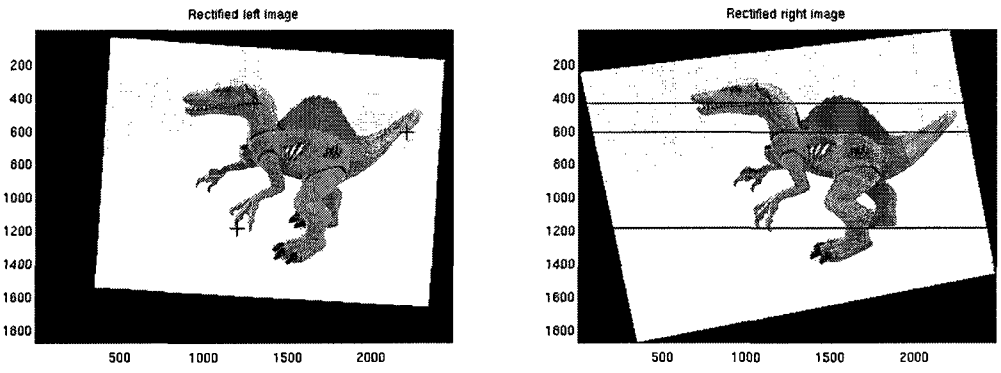


Figure 5-3. Rectified Dinosaur image pair.

The size of the rectified images is greater than the size of input images. For example, for the sample image pair of Dinosaur dataset which have been shown in Figure 5-2, the image size is 2000×1500, while the size of the rectified images which have been shown in Figure 5-3 is 2481×1881. The empty parts of the new image pair do not make any inconsistency because the rectified silhouette images will be used to fasten the stereo matching algorithm.

Using silhouette information decreases the computation time in two ways. By using the silhouette of the reference image, the number of pixels for which the matching search is processed is decreased. Inversely, silhouette information of target image decreases the number of pixels which are searched to find the best match. As it can be seen clearly in Figure 5-4, the number of foreground pixels is much less than the number of background ones.

After rectifying the image pairs, the depth maps are produced using stereo matching application. For matching measurement, I used normalized cross correlation which is a correlation measure.

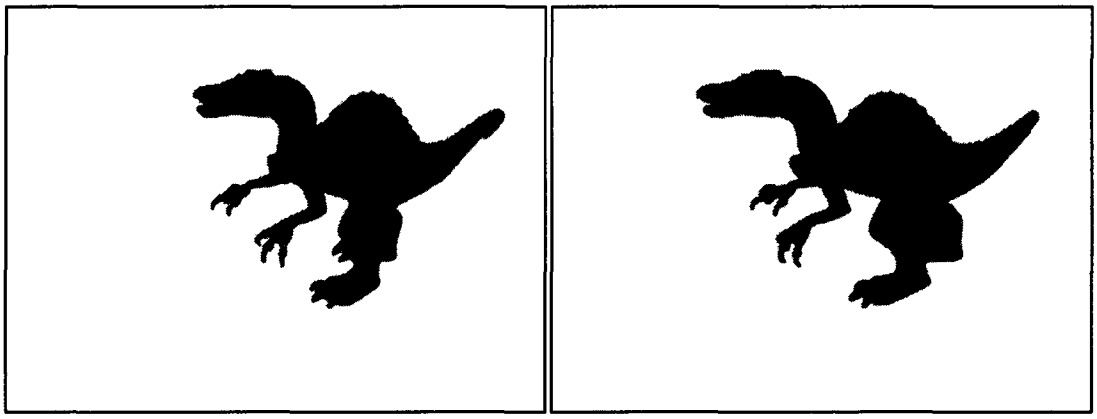


Figure 5-4. Rectified silhouette image pairs.

Because in my model, I need the reliable depth information, I applied the *Left-Right Consistency (LRC)* check to remove the inconsistent depth information from two directions. In left-right consistency check, first a depth map is produced based on the reference and target image pairs, which is referred as left-to-right depth map. Then

another depth map is produced based on the reference and target images as the new target and reference images, correspondingly. The second depth map is called right-to-left depth map. Figure 5-5 shows the resulted depth maps for the sample image pairs of Dinosaur dataset.

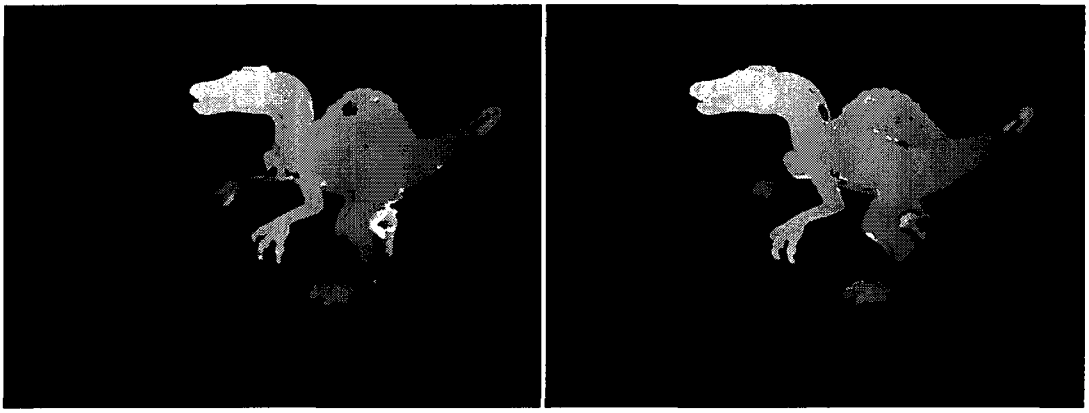


Figure 5-5. Left-to-right (left image) and right-to-left (right image) depth maps produced for the sample image pairs of Dinosaur dataset.

The last step of LRC check is to find the consistent disparity values between two produced depth maps. For each pixel of the left-to-right depth map, which is denoted as the reference pixel, the matched pixel of the right-to-left is selected, which is denoted as the target pixel. If the matched pixel for the target pixel of the target image is the reference pixel in the reference image, the depth value for the reference pixel is consistent from two views. Otherwise, the depth value is inconsistent and will be removed from the depth map.

Usually a threshold is used in LRC check, which defines the valid shifting between two views depth map. In my experiments, no threshold is used, and the depth values which are not exactly the same are removed. Figure 5-6 shows the resulted depth map after left-right consistency check. Comparison of the depth maps before and after the consistency check shows that the invalid depth values are removed, and the resulted depth values are reliable for the next steps of the algorithm. For example, the lighter pixels on the right leg of Dinosaur have invalid values because the left leg is nearer to the camera position. So they have been removed from the depth map.

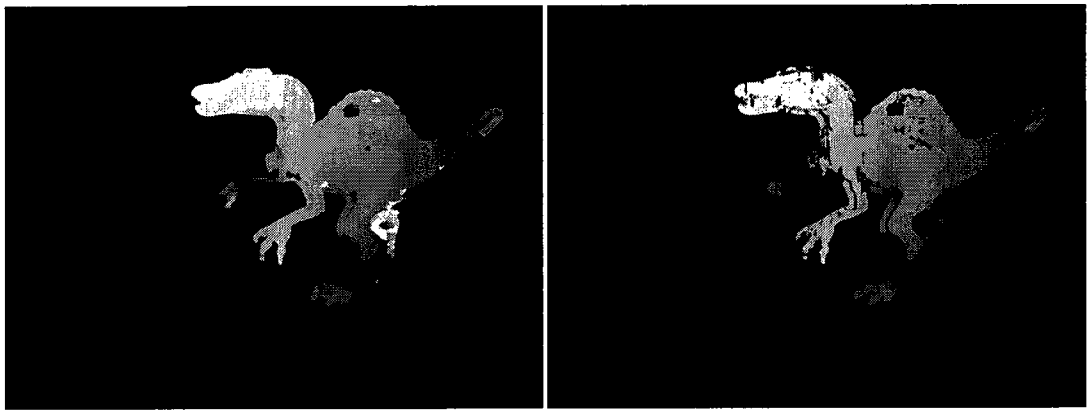


Figure 5-6. Left-to-right depth map (left image) and resulted depth map after LRC check (right image) for the sample image pairs of Dinosaur dataset.

Each pixel of the depth map has a depth value, which is zero if there is no reliable depth value. After producing the reliable depth maps, the resulted maps are back transformed to the image planes before rectification, which is called the *back rectification.* Back rectification produces two maps, the x-map and the y-map. Based on the x-map and y-map, the corresponding pixel for each pixel of the reference image is identified. Using a triangulation, the position of the 3D point corresponding for each pixel is determined. The triangulation is done using intersecting the 3D ray from the reference pixel of the reference image with the 3D ray from the corresponding pixel of the target image. The resulted 3D points for the first view of Dinosaur dataset have been shown in Figure 5-7.

Figure 5-7. Reconstructed 3D points for the first view of Dinosaur dataset.

As it can be seen clearly in this figure, there are many invalid 3D points which should be removed from the point cloud. Refining 3D points will be done in the next step over all the camera views.

5.3.2. Merging 3D Points

After producing the 3D points, based on the individual image pairs, the position of the 3D points will be refined using images from different views. Finally, all the 3D points from different viewpoints are combined to produce a point cloud. A surface fitting step provides a triangular mesh over the 3D point cloud, which is done to provide the appropriate type of results for Middlebury benchmark to evaluate the proposed model.

The first step of the merging process is to refine the position of the produced 3D points, followed by removing the inconsistent points. The position of the constructed 3D points is changed along its corresponding viewing ray to find the best position. For each viewpoint, k nearest cameras are selected. Then another window matching is done between the reference image and the k nearest camera images to measure the different positions of the 3D point.

This window matching process is a little bit different. For each pixel of the reference image, the position of the corresponding 3D point is moved along the viewing ray. For each new position, the 3D point is projected to the nearest cameras. The correlation value between the reference window and the window of the projected pixel is calculated for each nearest camera, using normalized cross correlation measurement. The overall correlation values for the new positions are the average NCC values over the k nearest cameras.

$$NCC(P) = \frac{\sum_{i=1}^{k} NCC\left(R(p), C_i(p_i')\right)}{k} \qquad (7)$$

where p and P denote the reference pixel and its corresponding 3D point with the new position, respectively. p_i' denotes the projection of the 3D point P into the camera i. R and

C_i denote the neighbor window of the points in reference image and the nearest camera i, correspondingly.

To have the better time performance, I first move position of the 3D point along the viewing ray by Δd to do a coarse refinement, and then move it near the coarse position by $\Delta d/10$ to do the final refinement.

The next step is to remove the refined 3D points which are inconsistent for all viewpoints. The inconsistency here is defined by the following criteria.

1. The projection of the point is mapped to the background segment of the silhouette image for at least one viewpoint.
2. The NCC value for the new position of the point is less than a threshold.
3. The distance between the point and the camera position is much greater or less than the average of the neighbor distances, using a threshold.

The points which have at least one of the mentioned criteria will be removed from the point cloud. Then for the pixels whose corresponding 3D point is removed a new 3D point from the point cloud which is mapped to those pixels will be selected. The inconsistency criteria are computed for the new 3D points. If the new 3D points are also inconsistent, they will be removed again without any substitutions. Figure 5-8 shows the refined 3D points for the first view of Dinosaur dataset. Comparison of Figure 5-7 and Figure 5-8 shows the quality of refinement.

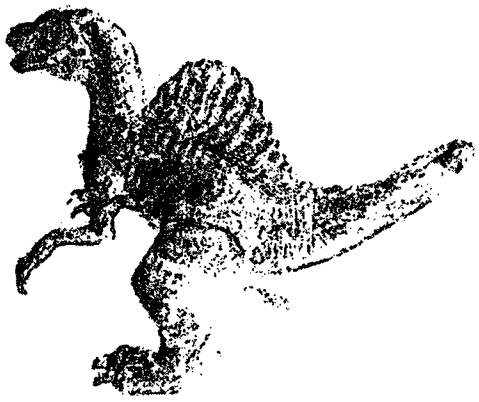


Figure 5-8. Refined 3D points for the first view of Dinosaur dataset.

The final step is to provide a surface mesh over the 3D points. Because Poisson surface reconstruction [65] is robust to the noisy data of the point cloud and non-uniformity distribution of the 3D points, it is used as the final step of the proposed model. Moreover, Poisson surface reconstruction produces a watertight mesh as a final result which is valuable for the Middlebury evaluation. The implementation code and binary executable version of the code is provided by the authors.

Because Poisson surface reconstruction gets an oriented point cloud as an input, the normal direction for each 3D point should be calculated. I used the information of the neighbor pixels in each viewpoint to estimate the normal direction for each 3D point. For each pixel, the normal directions of the planes which contain the corresponding 3D points of the pixel and two consecutive neighbors are calculated. The interesting normal direction for the 3D point is the average of calculated normal directions.

5.4. Experiments

For this proposed model, I applied the algorithm on two datasets, Dinosaur dataset from 3D Photography datasets and DinoSparseRing dataset from Middlebury benchmark. The results have been shown in the next subsections, followed by an evaluation subsection. For evaluating the results of my proposed model qualitatively, I sent the results to Middlebury College, and I got the evaluation results including accuracy and completeness. Based on these metrics, the proposed method is compared with the state-of-the-art approaches.

5.4.1. Results

The main results of the proposed model are the 3D surface points. For Dinosaur dataset, the results of each step of the proposed method have been shown in previous subsections. However, the final results which are the 3D surface points for Dinosaur dataset is presented in Figure 5-9. Figure 5-10 shows the 3D surface points resulted for DinoSparseRing dataset. The result of Dinosaur dataset looks denser than the result of

DinoSparseRing, because size of Dinosaur images is much more than DinoSparseRing image size, 2000×1500 for Dinosaur dataset versus 640×480 for DinoSparseRing dataset.

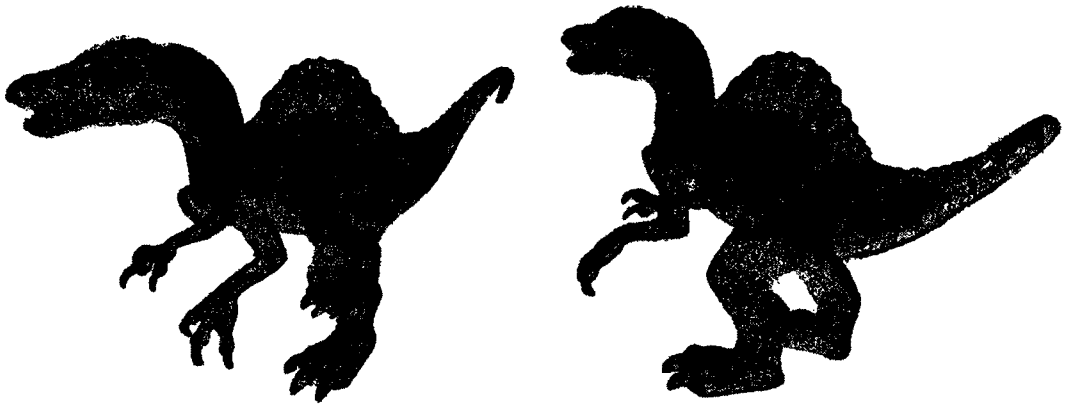


Figure 5-9. Resulted 3D surface points for Dinosaur dataset.

As it can be seen clearly in Figure 5-10 left, some parts of the Dino object is missing, since in one of the provided images of DinoSparseRing dataset, the mentioned parts are located out of the scope of the image. The inconsistent image is the second image of DinoSparseRing dataset with *dinoSR0002.png* filename, which has been shown in Figure 5-11. However, the proposed algorithm is designed to remove all the reconstructed parts of the 3D object which are inconsistent with at least one of the calibrated images.

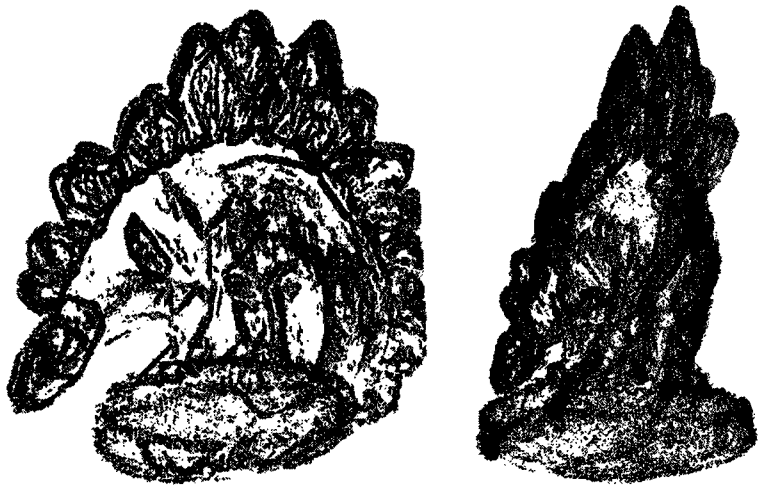


Figure 5-10. Resulted 3D surface points for DinoSparseRing dataset.

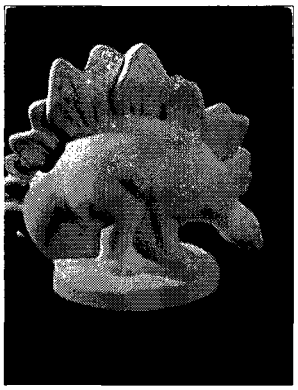


Figure 5-11. The inconsistent image of DinoSparseRing dataset.

To evaluate the proposed method on Middlebury benchmark, the result should be submitted to Middlebury College as a triangular surface mesh. As it mentioned in the steps of the proposed algorithm, I used Poisson Surface Reconstruction to provide the surface over the resulted 3D surface points. The 3D triangular final result for DinoSparseRing dataset has been shown in Figure 5-12 from different viewpoints.

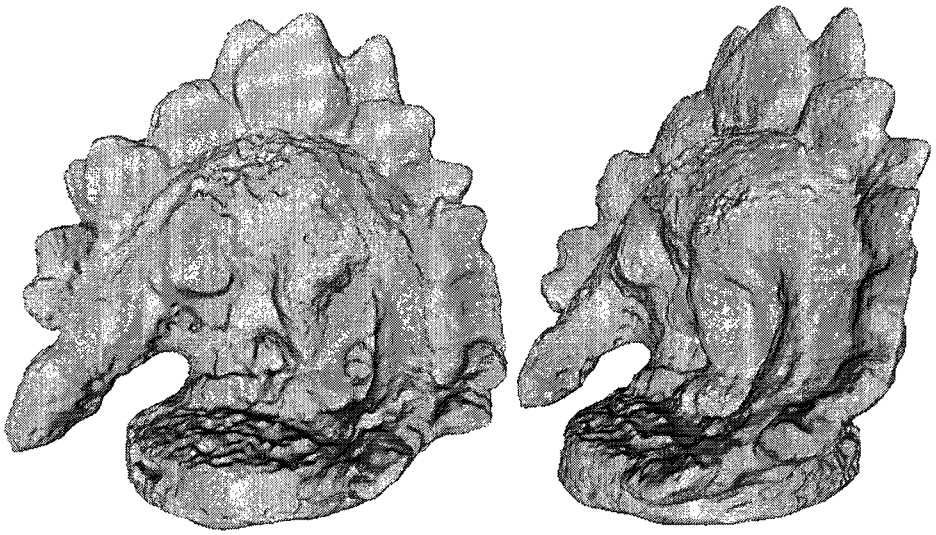


Figure 5-12. Final 3D triangular mesh for DinoSparseRing dataset.

Two other views of the final triangular mesh are presented in Figure 5-13.

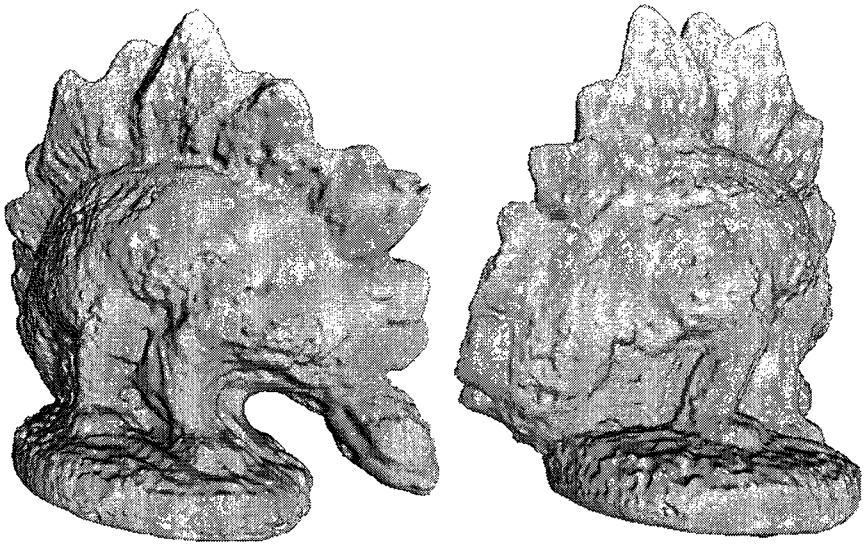


Figure 5-13. Final 3D triangular mesh for DinoSparseRing dataset from different views.

5.4.2. Comparison and Evaluation

The final 3D triangular surface mesh for the DinoSparseRing dataset has been sent to Middlebury College for evaluation. The evaluation results are available online [70]. The result of the proposed visual hull model is also evaluated by Middlebury benchmark. On Middlebury evaluation webpage, the evaluation of the proposed models is usually denoted by the last name of the main author. So, my proposed visual hull model is denoted by *Raeesi*, and the proposed 3D object reconstruction model is denoted by *Raeesi2*.

Table 5-1 shows the evaluation of both models for DinoSparseRing dataset. The values mentioned for accuracy are the fraction threshold, and the values determined for completeness are the distance threshold. It means, for example, 90 percent of the surface points of the visual hull result are within the distance 4.84*mm* of the ground truth and 38.8% of the ground truth surface points are within distance 1.25*mm* of the visual hull result.

Table 5-1. Evaluation results of both proposed models on Middlebury benchmark for DinoSparseRing dataset. The accuracy is in millimeter and completeness is percentage.

Proposed Models	Accuracy		Completeness	
	90%	80%	1.25*mm*	0.75*mm*
Visual Hull Model	4.84	3.44	38.8	19.0
3D Object Reconstruction	0.63	0.42	95.0	86.0

As it was expected, the result of the visual hull model is much coarser than the 3D object reconstruction model. Table 5-1 shows that the accuracy of the 3D reconstruction model is 8 times more than the accuracy of the visual hull model.

The comparison of the result of proposed 3D object reconstruction model is compared with the state-of-the-art models in two steps. The first one is the overall comparison which compares all the models, regardless of their category. The second one compares my model with the modern depth map based approaches.

It should be mentioned that the name presented in the following tables are the same as the names displayed on Middlebury evaluation webpage, which is usually the last name of the author. In some cases, the methods are denoted by anonymous, but the title of submitted paper and the corresponding conference or journal is determined. These cases are called *Submitted*. Besides, the category of a model is unknown, because there is no information about the proposed algorithm.

Table 5-2 and Table 5-3 show the accuracy and completeness results of all the state-of-the-art models. The result of my proposed model has been shaded in both tables. My model obtained rank 16 out of 28 for accuracy where its fraction threshold is 90% and rank 17 for completeness where the its distance threshold is 1.25*mm*. For both accuracy and completeness metric, the top performer is Furukawa3 which is proposed by Y. Furukawa and J. Ponce [45].

Table 5-2. Comparison of accuracy of the state-of-the-art models. The accuracy threshold is 90%.

No	Algorithms	Year	Category	Accuracy
1	Furukawa3 [45]	2008	Feature Extraction	0.37
2	Bradley [54]	2008	Depth Map Based	0.38
3	ECCV_216 [60]	Submitted	Depth Map Based	0.42
4	Furukawa2 [44]	2007	Feature Extraction	0.42
5	Deng [59]	Submitted	Depth Map Based	0.43
6	Zaharescu [36]	2007	Surface Evolution	0.45
7	Kun Li [58]	Submitted	Depth Map Based	0.47
8	ECCV_642 [42]	Submitted	Surface Evolution	0.48
9	Liu2 [57]	2009	Depth Map Based	0.51
10	Kolev2 [37]	2009	Surface Evolution	0.53
11	Song [32]	Submitted	3D Volumetric	0.54
12	Goesele [56]	2006	Depth Map Based	0.56
13	Furukawa [34]	2006	Surface Evolution	0.58
14	Liu [55]	2009	Depth Map Based	0.59
15	Hernandez [35]	2004	Surface Evolution	0.60
16	Raeesi2	Proposed	Depth Map Based	0.63
17	Jancosek-3DIM09 [47]	2009	Feature Extraction	0.66
18	SurfEvolution [43]	Submitted	Surface Evolution	0.66
19	Pons [39]	2005	Surface Evolution	0.71
20	Auclair [33]	2008	Surface Evolution	0.74
21	Gargallo [50]	2007	Surface Evolution	0.76
22	Delaunoy [49]	2008	Surface Evolution	0.89
23	Ladikos [38]	2008	Surface Evolution	0.89
24	Starck [48]	2007	Feature Extraction	1.01
25	NIPS_829 [61]	Submitted	-	1.07
26	Vogiatzis [41]	2005	Surface Evolution	1.18
27	Tran [40]	2006	Surface Evolution	1.26
28	Strecha [51]	2006	Surface Evolution	1.41

Table 5-3. Comparison of completeness of the state-of-the-art models. The completeness threshold is 1.25mm.

No	Algorithms	Year	Category	Completeness
1	Furukawa 3 [45]	2008	Feature Extraction	99.2
2	Furukawa 2 [44]	2007	Feature Extraction	99.2
3	Zaharescu [36]	2007	Surface Evolution	99.2
4	Liu2 [57]	2009	Depth Map Based	98.7
5	ECCV_642 [42]	Submitted	Surface Evolution	98.6
6	Hernandez [35]	2004	Surface Evolution	98.5
7	Kolev2 [37]	2009	Surface Evolution	98.3
8	Liu [55]	2009	Depth Map Based	98.3
9	ECCV_216 [60]	Submitted	Depth Map Based	97.8
10	Deng [59]	Submitted	Depth Map Based	97.8
11	Pons [39]	2005	Surface Evolution	97.7
12	SurfEvolution [43]	Submitted	Surface Evolution	97.6
13	Kun Li [58]	Submitted	Depth Map Based	97.4
14	Furukawa [34]	2006	Surface Evolution	96.9
15	Auclair [33]	2008	Surface Evolution	96.8
16	Song [32]	Submitted	3D Volumetric	95.5
17	Raeesi2	Proposed	Depth Map Based	95.0
18	Ladikos [38]	2008	Surface Evolution	95.0
19	Bradley [54]	2008	Depth Map Based	94.7
20	Delaunoy [49]	2008	Surface Evolution	93.9
21	Strecha [51]	2006	Surface Evolution	91.5
22	NIPS_829 [61]	Submitted	-	91.0
23	Vogiatzis [41]	2005	Surface Evolution	90.8
24	Gargallo [50]	2007	Surface Evolution	90.7
25	Starck [48]	2007	Feature Extraction	90.7
26	Tran [40]	2006	Surface Evolution	89.3
27	Jancosek-3DIM09 [47]	2009	Feature Extraction	74.9
28	Goesele [56]	2006	Depth Map Based	26.0

By changing the threshold, the ranking will be changed. The best rank of my model for accuracy is 13 with threshold 99% and the best rank for completeness is 17 with distance thresholds 1.25, 1.5, 1.75, and 2*mm*.

As it can be seen clearly, the results are so close to each other, such that the difference less than 0.1mm can change the rank of a model by many steps. However, to show the small differences more clearly, all the comparisons in this section are presented in bar charts in Appendix B.

The next comparisons are among the depth map based models. Table 5-4 shows the comparison of the depth map based models for DinoSparseRing dataset. Considering accuracy metric, the best method is Bradley which is published by D. Bradley et al. [54] in 2008. The most recent approaches do not obtain better accuracy than Bradley accuracy. However the rank of my model is one of the latest ranks, the obtained accuracy is acceptable and comparable with the published models.

Table 5-4. Comparison of accuracy of the state-of-the-art depth map based models. The accuracy threshold is 80%.

No	Algorithms	Year	Accuracy
19	Bradley [54]	2008	0.27
9	ECCV_216 [60]	Submitted	0.27
10	Deng [59]	Submitted	0.30
13	Kun Li [58]	Submitted	0.34
28	Goesele [56]	2006	0.36
4	Liu2 [57]	2009	0.36
17	Raeesi2	Proposed	0.42
8	Liu [55]	2009	0.47

The comparison of completeness metric has been shown in Table 5-5. The top performer approach in case of completeness is Liu2 which is proposed by Y. Liu et al. [57]. Like accuracy comparison, this comparison shows that the new proposed methods do not obtained better completeness than Liu2 completeness. However, my proposed model obtains an acceptable completeness as well. The most interesting issue between

Table 5-4 and Table 5-5 is that for completeness metric the rank of Bradley method which is the top performer in case of accuracy is one of the latest ranks. In opposite cases, Liu and Liu2 obtain the best completeness, while their accuracy results are the worst among the modern approaches.

Table 5-5. Comparison of completeness of the state-of-the-art depth map based models. The completeness threshold is 1.5*mm*.

No	Algorithms	Year	Completeness
4	Liu2 [57]	2009	99.4
8	Liu [55]	2009	99.0
10	Deng [59]	Submitted	98.9
13	Kun Li [58]	Submitted	98.6
9	ECCV_216 [60]	Submitted	98.4
17	Raeesi2	Proposed	96.8
19	Bradley [54]	2008	95.0
28	Goesele [56]	2006	26.1

Among the published depth map based methods, only Liu2 has better results for both accuracy and completeness metrics than my proposed model. However, the comparison shows that the proposed model obtains an acceptable accuracy as well as an acceptable completeness which are comparable with the modern approaches in this field.

6. Conclusions

There are many applications from different areas which need to localize, recognize and reconstruct the 3D objects. The desired quality of the resulted 3D shape of the object and the acceptable time performance of the reconstruction process depends only on the applications. Some applications need to reconstruct a coarse shape of the object in acceptable time. In robotics, for example, it is very important for robot to find the positions of the obstacle at the real time, while the quality of the reconstructed shape of the object does not matter. In contrast, some other applications need to reconstruct the object as accurate as possible, for which the time performance is not important. For instance, in inverse engineering, the goal is to provide an accurate model of the existing object.

Vision network is one of the solutions for all of these applications. Generally, vision network is one of the cheapest existing solutions, which is easily configurable. Moreover, vision networks are able to reconstruct the 3D shape of the object in different level of details within different amount of time. It captures some images from different views, and produces the 3D shape of the object. The first issue in vision network is camera calibration, which can be done as an automatic process in network configuration step.

The coarsest, while fastest model of the vision network models is the convex hull of the object. So, convex hulls can be the solution for the real time applications such as obstacle avoidance. Reconstruction of the accurate 3D shape of the object, which is called 3D object reconstruction, requires much amount of time. In 3D modeling, for instance, the accuracy of the model is very important, and the goal of the application is to produce a model as accurate as possible, while the time performance does not matter. So, the 3D object reconstruction can be the solution for these applications.

Like in the other fields, there is a trade-off between time and accuracy for reconstructing the 3D shape of the object. Visual hull model is a model which produces

an acceptable shape of the object in acceptable time. However, it depends on the application to select the best existing models in the area of vision networks.

In this thesis, I proposed two different models, a visual hull model and a 3D object reconstruction model. For the visual hull model, the contribution is to provide the bounding surfaces over the bounding edge model of the object, and merging them. Because bounding edge model is one of the fastest visual hull models, the proposed model is faster than most of the existing approaches. The evaluation of the results of the proposed visual hull model has been describes in section 4. Moreover, the proposed method can be computed in distributed manner. In distributed computing, the execution time is divided to the number of views, which increases the time performance of the reconstruction, dramatically.

The proposed 3D object reconstruction model is a depth map based model, which produces the 3D points for each viewpoint, and merges them to a point cloud. At the final step, it fits a triangular surface mesh over the refined 3D point cloud. The evaluation of the results shows that the proposed model obtains an acceptable accuracy as well as acceptable completeness which are comparable with existing approaches.

7. References

[1]. K.M. Cheung, S. Baker, and T. Kanade. Shape-From-Silhouette Across Time Part II: Applications to Human Modeling and Markerless Motion Tracking. International Journal of Computer Vision, 63(3):225 – 245, August 2005.

[2]. Middlebury Multi-View Datasets. Middlebury College, Microsoft Research, and the National Science Foundation. Available at http://vision.middlebury.edu/mview/.

[3]. W. N. Martin, and J. K. Aggarwal. Volumetric Description of Objects from Multiple Views. IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI), 5(2): 150-158, 1983.

[4]. Aaron Mavrinac. Feature-Based Calibration of Distributed Smart Stereo Camera Networks. Master of Applied Science Thesis, Electrical and Computer Engineering Department, University of Windsor, May, 2008.

[5]. Hamid Aghajan, Chen Wu, and Richard Kleihorst. Distributed Vision Networks for Human Pose Analysis. book chapter, March 23, 2008.

[6]. B.G. Baumgart. A Polyhedron Representation for Computer Vision. In Proc. of AFIPS National Computer Conference, 1975.

[7]. Y. Furukawa, and J. Ponce. 3D Photography Dataset. Beckman Institute and Department of Computer Science, University of Illinois at Urbana-Champaign. Available at http://www-cvr.ai.uiuc.edu/ponce_grp/data/mview/.

[8]. A. R. Smith, and J. F. Blinn. Blue Screen Matting. In Proc. of Computer Graphics (SIGGRAPH 96), pp. 21-30, August, 1996.

[9] M. Bichsel. Segmenting Simply Connected Moving Objects in a Static Scene. IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI), 16(11): 1138-1142, November, 1994.

[10]. A. Laurentini. The Visual Hull: A New Tool for Contour-Based Image Understanding. In Proc. of 7th Scandinavian Conf. Image Analysis, pp. 993-1002, 1991.

[11]. K. Cheung, S. Baker, and T. Kanade. Shape-From-Silhouette Across Time Part I: Theory and Algorithms. International Journal on Computer Vision, 62(3): 221 – 247, May 2005.

[12]. A. Laurentini. The Visual Hull Concept for Silhouette-Based Image Understanding. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 16(2):150–162, February 1994.

[13]. J.S. Franco and E. Boyer. Exact Polyhedral Visual Hulls. In Fourteenth British Machine Vision Conference (BMVC), pp. 329–338, Norwich, UK, September 2003.

[14]. C. L. Jackins and S. L. Tanimoto. Oct-trees and Their Use in Representing Three-Dimensional Objects. In Proc. of Computer Graphics and Image Processing, 14: 249-270, 1980.

[15]. C. H. Chien and J. K. Aggarwal. Volume/Surface Octrees for the Representation of Three-Dimensional Objects. In Proc. of Computer Vision, Graphics, and Image Processing archive. 36: 100-113, 1986.

[16]. R. Szeliski. Rapid Octree Construction from Image Sequences. CVGIP, 58(1):23–32, 1993.

[17]. K. Kutulakos and S. Seitz. A Theory of Shape by Space Carving. International Journal of Computer Vision, 38(3):199–218, 2000.

[18]. G. Cheung. 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.

[19]. S. Lazebnik, E. Boyer, and J. Ponce. On Computing Exact Visual Hulls of Solids Bounded by Smooth Surfaces. In Proc. of Computer Vision and Pattern Recognition (CVPR), Dec. 2001.

[20]. C. Buehler,W.Matusik, and L.McMillan. Polyhedral Visual Hulls for Real-Time Rendering. In Proceedings of Eurographics Workshop on Rendering, 2001.

[21]. C. Buehler, W. Matusik, L. McMillan, and S. Gortler. Creating and Rendering Image-Based Visual Hulls. Technical Report MIT-LCS-TR-780, MIT, 1999.

[22]. W. Matusik, C. Buehler, R. Raskar, S. J. Gortler, and L. McMillan. Image-Based Visual Hulls. In Proc. of Computer Graphics (SIGGRAPH), July 2000.

[23]. G. Cheung, S. Baker, and T. Kanade. Visual Hull Alignment and Refinement Across Time: A 3D Reconstruction Algorithm Combining Shape-From-Silhouette with stereo. In Proc. of Computer Vision and Pattern Recognition (CVPR), Madison, MI, 2003.

[24]. Middlebury Stereo Vision Datasets. Middlebury College, Microsoft Research, and the National Science Foundation. Available at http://vision.middlebury.edu/stereo/.

[25]. S. Lazebnik, Y. Furukawa, and J. Ponce. Projective Visual Hulls. International Journal of Computer Vision, 74(2): 137 – 165, August 2007.

[26]. MeshLab software. Available at http://meshlab.sourceforge.net/

[27]. The Stanford 3D Scanning Repository. Available at http://www-graphics.stanford.edu/data/3Dscanrep/.

[28]. Intel's OpenCV library written in C programming language. Available at http://sourceforge.net/projects/opencvlibrary/.

[29]. Camera Calibration Toolbox for Matlab. Available at http://www.vision.caltech.edu/bouguetj/calib_doc/.

[30]. S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski. A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms. In Proc. of Computer Vision and Pattern Recognition (CVPR), 1: 519-526, 2006.

[31]. S. Seitz and C. Dyer. Photorealistic Scene Reconstruction by Voxel Coloring. IJCV, 35(2):151.173, 1999.

[32]. P. Song, X. Wu, and M. Wang. Volumetric Stereo and Silhouette Fusion for Image-Based Modeling. Submitted to the Visual Computer Journal, 2010.

[33]. A. Auclair, L. Cohen, and N. Vincent. Using Point Correspondences without Projective Deformation for Multi-View Stereo Reconstruction. In Proc. of International Conference on Image Processing (ICIP), pp. 193-196, 2008.

[34]. Y. Furukawa and J. Ponce. High-Fidelity Image-Based Modeling. Technical Report 2006-02, UIUC, 2006.

[35]. C. Hernandez Esteban and F. Schmitt. Silhouette and Stereo Fusion for 3D Object Modeling. Computer Vision and Image Understanding (CVIU), 96(3):367-392, 2004.

[36]. A. Zaharescu, E. Boyer, and R. Horaud. TransforMesh: A Topology-Adaptive Mesh-Based Approach to Surface Evolution. Asian Conference on Computer Vision (ACCV), 2: 166-175, 2007.

[37]. K. Kolev, M.Klodt, T. Brox, and D. Cremers. Continuous Global Optimization in Multi-View 3D Reconstruction. International Journal on Computer Vision (IJCV), 84(1): 80-96, 2009.

[38]. A. Ladikos, S. Benhimane, and N. Navab. Multi-View Reconstruction Using Narrow-Band Graph-Cuts and Surface Normal Optimization. In Proc. of British Machine Vision Conference (BMVC), 2008.

[39]. J.-P. Pons, R. Keriven, and O. Faugeras. Modelling Dynamic Scenes by Registering Multi-View Image Sequences. In Proc. of Computer Vision and Pattern Recognition (CVPR), 2: 822-827, 2005.

[40]. S. Tran and L. Davis. 3D Surface Reconstruction Using Graph Cuts with Surface Constraints. In Proc. of European Conference on Computer Vision (ECCV), 2: 219-231, 2006.

[41]. G. Vogiatzis, P. Torr, and R. Cipolla. Multi-View Stereo via Volumetric Graph-Cuts. In Proc. of Computer Vision and Pattern Recognition (CVPR), pp. 391-398, 2005.

[42]. Anonymous. Anisotropic Minimal Surfaces Integrating Photoconsistency and Normal Information for Multiview Stereo. Submitted to European Conference on Computer Vision (ECCV), submission 642, 2010.

[43]. Anonymous. An Iterative Surface Evolution Algorithm for Multiview Stereo. Submitted to EURASIP Journal on Image and Video Processing (EURASIP JIVP).

[44]. Y. Furukawa and J. Ponce. Accurate, Dense, and Robust Multi-View Stereopsis. In Proc. of Computer Vision and Pattern Recognition (CVPR), pp. 1-8, 2007.

[45]. Y. Furukawa and J. Ponce. Accurate, Dense, and Robust Multi-View Stereopsis. IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI) 2008.

[46]. C. Harris and M. J. Stephens. A Combined Corner and Edge Detector. In Alvey88, pp. 147-152, 1988.

[47]. M. Jancosek, A. Shekhovtsov, and T Pajdla. Scalable Multi-View Stereo. IEEE International Workshop on 3D Digital Imaging and Modeling (3DIM), 2009.

[48]. J. Starck and A. Hilton. Surface Capture for Performance-Based Animation. In Proc. of Computer Graphics and Applications (CG&A), 27(3):21-31, 2007.

[49]. A. Delaunoy, E. Prados, P. Gargallo, J.-P. Pons and P. Sturm. Minimizing the Multi-view Stereo Reprojection Error for Triangular Surface Meshes. In Proc. of British Machine Vision Conference (BMVC) 2008.

[50]. P. Gargallo, E. Prados, and Peter Sturm. Minimizing the Reprojection Error in Surface Reconstruction from Images. In Proc. of International Conference on Computer Vision (ICCV), pp. 1-8, 2007.

[51]. C. Strecha, R. Fransens, and L. Van Gool. Combined Depth and Outlier Estimation in Multi-View Stereo. In Proc. of Computer Vision and Pattern Recognition (CVPR), 2: 2394-2401, 2006.

[52]. R. Szeliski. A Multi-View Approach to Motion and Stereo. In Proc. of Computer Vision and Pattern Recognition (CVPR), 1:157-163, 1999.

[53]. P. Gargallo and P. Sturm. Bayesian 3D Modeling from Images Using Multiple Depth Maps. In Proc. of Computer Vision and Pattern Recognition (CVPR), 2:885-891, 2005.

[54]. D. Bradley, T. Boubekeur, and W. Heidrich. Accurate Multi-View Reconstruction Using Robust Binocular Stereo and Surface Meshing. In Proc. of Computer Vision and Pattern Recognition (CVPR), 2008.

[55]. Y. Liu, Q. Dai and W. Xu. A Point Cloud Based Multi-View Stereo Algorithm for Free Viewpoint Video. IEEE Transactions on Visualization and Computer Graphics, 16(3): 407-418, 2009.

[56]. M. Goesele, B. Curless, and S. M. Seitz. Multi-View Stereo Revisited. In Proc. of Computer Vision and Pattern Recognition (CVPR), pp. 2402-2409, 2006.

[57]. Y. Liu, X. Cao, Q. Dai, and W. Xu. Continuous Depth Estimation for Multi-View Stereo. In Proc. of Computer Vision and Pattern Recognition (CVPR), pp. 2121-2128, 2009.

[58]. K. Li, Q. Dai, and W. Xu. Markerless Shape and Motion Capture from Multi-View Video Sequences. Submitted to IEEE Transactions on Circuits and Systems for Video Technology, 2009.

[59]. Y. Deng, Y. Liu, Q. Dai, and Z. Zhang. Depth Maps Fusion for Multi-View Stereo via Matrix Completion. Submitted to IEEE Transaction on Pattern Analysis and Machine Intelligence (PAMI), 2010.

[60]. Anonymous. An Integrated Depth Fusion Approach for Multi-View Stereo. Submitted to European Conference on Computer Vision (ECCV), submission 216, 2010.

[61]. Anonymous. Ray-view Markov Random Fields for Image-Based 3D Modeling: Model and Efficient Inference. Submitted to Neural Information Processing Systems (NIPS) submission 829, 2009.

[62]. M. Berg, O. Cheong, M. Kreveld, M. Overmars. Computational Geometry: Algorithms and Applications. Springer-Verlag, ISBN 978-3-540-77973-5, 2008. Available at http://www.cs.uu.nl/geobook/interpolation.pdf.

[63]. R. Kolluri, J. Shewchuk, J. O'Brien. Spectral Surface Reconstruction from Noisy Point Clouds. Symposium on Geometry Processing, pp. 11–21, 2004.

[64]. H. Hoppe, T. Derose, T. Duchamp, J. McDonald, W. Stuetzle. Surface Reconstruction from Unorganized Points. In Proc. of Computer Graphics (SIGGRAPH), 26: 71–78, 1992.

[65]. M. Kazhdan, M. Bolitho, and H. Hoppe. Poisson Surface Reconstruction. Eurographics Symposium on Geometry Processing, pp. 61–70, 2006.

[66]. Poisson Surface Reconstruction Implementation. Available at http://www.cs.jhu.edu/~misha/Code/PoissonRecon/.

[67]. B. Curless, M. Levoy. A Volumetric Method for Building Complex Models from Range Images. In Proc. of Computer Graphics (SIGGRAPH), pp. 303-312, 1996.

[68]. Vrippack: Volumetric Range Image Processing Package. Available at http://grail.cs.washington..edu/software-data/vrip/.

[69]. A. Fusiello, E. Trucco, and A. Verri. A Compact Algorithm for Rectification of Stereo Pairs. Machine Vision and Applications, 12(1):16–22, 2000.

[70]. Evaluation Results of My Proposed Models. Available at http://vision.middlebury.edu/mview/eval/doAuth.php?aid=raeesi2-may-13-2010.

Appendix A: Implementation using Java

The main part of the proposed models is implemented in Java programming language. I implemented 31 classes in a single package which is called *VisualHull*. As it mentioned in the main context, there is a metric of the size of implemented application which is the source lines of code. There are two type of LOC, physical and logical. Physical line of code metric counts number of lines in the source code file which includes the lines that have statements as well as blank line and comment line, while logical line of code metric counts only statements.

Table A-1 shows the implemented class files in Java in alphabetical order, with their line of code metrics and number of static and non-static methods. There are two main class files, one for each proposed model. The functions of each class file are described as follows.

1. **Camera**: The camera object models a camera with its intrinsic and extrinsic parameters which determine its position and direction. Each camera has a captured image with its corresponding silhouette.
2. **Contour**: This object finds the contour of a silhouette image, which is used to calculate the bounding edges. The contour pixels are traversed in counterclockwise order, in which the map of the object is always at the left hand side of the direction of traversing the contour points.
3. **ContourElement**: The elements of Contour objects are modeled as ContourElement which are ordered silhouette pixels with the next and previous pointers.
4. **Coordinate2D**: The image and plane coordinates are modeled in the same way with one different factor which is the fact that the image coordinates are integer numbers.
5. **Coordinate3D**: The world and camera coordinates are implemented in Coordinate3D class. This class implemented all the functions needed for

3D vector computations such as addition, subtraction, dot production, cross production, and normalization.

Table A-1. Class files implemented in Java with number of physical and logical lines of code as well as number of implemented methods.

No	Class File	Physical LOC	Logical LOC	Number of Methods
1	Camera	671	394	52
2	Contour	198	160	10
3	ContourElement	68	56	14
4	Coordinate2D	215	115	24
5	Coordinate3D	242	138	20
6	Curves	20	15	3
7	DepthCurve	66	24	4
8	DepthMap	293	223	8
9	DepthMapImage	731	524	33
10	DistortionCoefficients	68	44	5
11	ExtrinsicParameters	192	133	15
12	FocalLength	42	24	4
13	Interval1D	313	211	16
14	Interval1DPoint	62	56	11
15	IntrinsicParameters	136	87	8
16	MainClass	361	234	9
17	MainClassDepthMap	360	210	8
18	MatchingDriver	113	77	11
19	MatlabControl	279	155	17
20	Mesh	312	274	7
21	Parameters	41	27	0
22	Point3D	110	102	27
23	PrincipalPoint	42	24	4
24	Project3Dray	321	239	3
25	Ray2D	139	52	11
26	Ray3D	128	67	11
27	RayIntervals	62	25	5
28	RotationMatrix	162	111	12
29	SkewCoefficient	37	17	4
30	TranslationMatrix	77	43	7
31	VisualHull	519	391	24
Total		**6380**	**4252**	**387**

6. **Curves**: Curves object is used to save the DepthCurve objects as a stack to manage Matlab calls.
7. **DepthCurve**: To find the best match for each pixel of the depth maps, all the correlation values are store as a curve which is implemented in DepthCurve class.
8. **DepthMap**: The calculated depth maps in Matlab are imported to Java using DepthMap object. Each object has a reference view, a target view, and two depth maps, x-map and y-map which contain the number of row and column of the best match pixel of the target image. Reference and target views are instances of the Camera object.
9. **DepthMapImage**: 3D points creation, and refinement are implemented as the functions of DepthMapImage object. In this object, first based on each viewpoints, 3D points are generated, followed by position refinement and removing inconsistent points.
10. **DistortionCoefficients**: DistortionCoefficients object models the distortion coefficients for each camera which is one of the camera intrinsic parameters. It is implemented as a vector of five coefficients.
11. **ExtrinsicParameters:** Camera extrinsic parameters for each camera are modeled in the object of ExtrinsicParameters including the rotation and translation matrices. In this object, transformations from world space to camera space, from camera space to image plane and vice versa are implemented.
12. **FocalLength**: The focal length of the cameras is modeled in FocalLength object, one of camera intrinsic parameters.
13. **Interval1D**: Bounding edges are implemented as the one dimensional intervals for each pixel. Interval1D object is the bounding edge with the start point and finish point.
14. **Interval1DPoint**: Each Interval1D object has two Interval1DPoint objects which are the start point and finish point of the corresponding bounding edge. Each point is stored as distance from 3D point to camera position.

15. **IntrinsicParameters**: Camera intrinsic parameters are implemented in IntrinsicParameters object which contains focal length, principal point, skew and distortion coefficients. Image plane to image pixel, and image pixel to image plane transformations are implemented in this object.
16. **MainClass**: The main class for proposed visual hull model is implemented in MainClass object. First, it starts by reading the camera information and their corresponding silhouette images, and finally at the last step, it calls the write function to write the final results to a file with PLY format.
17. **MainClassDepthMap**: MainClassDepthMap is the main class for the proposed 3D object reconstruction method which is a depth map based approach. It gets the camera information as well as the silhouette and color images and depth map information, and returns a text file which includes the information of the 3D points including their positions and normal directions.
18. **MatchingDriver**: The interface between Java and Matlab is implemented in Matching Driver object. The only Matlab called is considered for the implementation is the calculation of normalized cross correlation between two windows.
19. **MatlabControl**: To call Matlab from Java to do some function, MatlabControl object is used, which translates the functions of the interface to the Matlab codes.
20. **Mesh**: The Mesh object is implemented to produce the bounding surface meshes as well as the final mesh for the proposed visual hull model. It generates the triangular meshes as the PLY files.
21. **Parameters**: All the parameters of the proposed methods are stored in Parameters object including the color of the background and foreground pixels of the silhouettes, number of k for the nearest cameras for each depth map, the window size for stereo window matching, the configuration of the Matlab calls, the normalized cross correlation threshold, the interesting resolution for the results, and so on.

22. **Point3D**: The 3D points in implementation are modeled by the Point3D object, which includes position, normal direction, view direction (direction to the viewing camera), neighbor information and pointers to the 3D points which are mapped to its corresponding reference pixel.
23. **PrincipalPoint**: The PrincipaclPoint object contains the information of the camera principal point which is one of the intrinsic parameters.
24. **Project3DRay**: The projection of the 3D ray to the silhouette images is implemented as a static function in Project3Dray object, which returns the intersection parts of the 3D ray with the corresponding silhouette. It first project the 3D ray to a 2D ray in silhouette image plane, and then calculates the intersection of the 2D ray with the foreground pixels of the silhouette. Finally, it back-projects the 2D segments of the 2D ray to the 3D space.
25. **Ray2D**: The 2D rays are implemented as Ray2D object which is determined with the position of the start point of the ray and its direction. The position and direction are 2D plane coordinates.
26. **Ray3D**: The 3D rays are modeled in Ray3D object. Like the 2D ray, 3D rays are determined by their position and direction, while their position and direction are 3D space coordinates. The new instances of Ray3D object is calculated based on a pixel of an image, for which the start point is the position of the corresponding camera and direction is such that the ray goes through the 3D position corresponding to the interesting pixel in image plane.
27. **RayIntervals**: The information of the intersected rays for each contour pixel is tracked using RayIntervals, which is a collection of Interval1D objects.
28. **RotationMatrix**: The rotation matrix of the cameras is one of the extrinsic parameters which implemented in RotationMatrix object. Rotation matrix is a 3×3 matrix with floating values.
29. **SkewCoefficient**: Skew coefficient is one of camera intrinsic parameters which is implemented in SkewCoefficient object.

30. **TranslationMatrix**: Translation matrix, one of the camera extrinsic parameters, is a 3×1 matrix which determines the translation of the camera coordinate with respect to the space coordinate.
31. **VisualHull**: VisualHull object implements the main part of the proposed visual hull model. It keeps the bounding edge information as the 2D samples.

Appendix B: Comparison of the 2nd Proposed Model

The comparisons of the results of the proposed 3D object reconstruction model, which are discussed in Section 5, are presented in the following bar charts. Figure B-1 shows the comparison of the proposed method with all the state-of-the-art approaches for three different thresholds, 80%, 90%, and 99%. As it can be seen clearly, the accuracy values are so close to each other. The methods are sorted based on the accuracy for threshold 99% which is almost all the points of the result surface mesh.

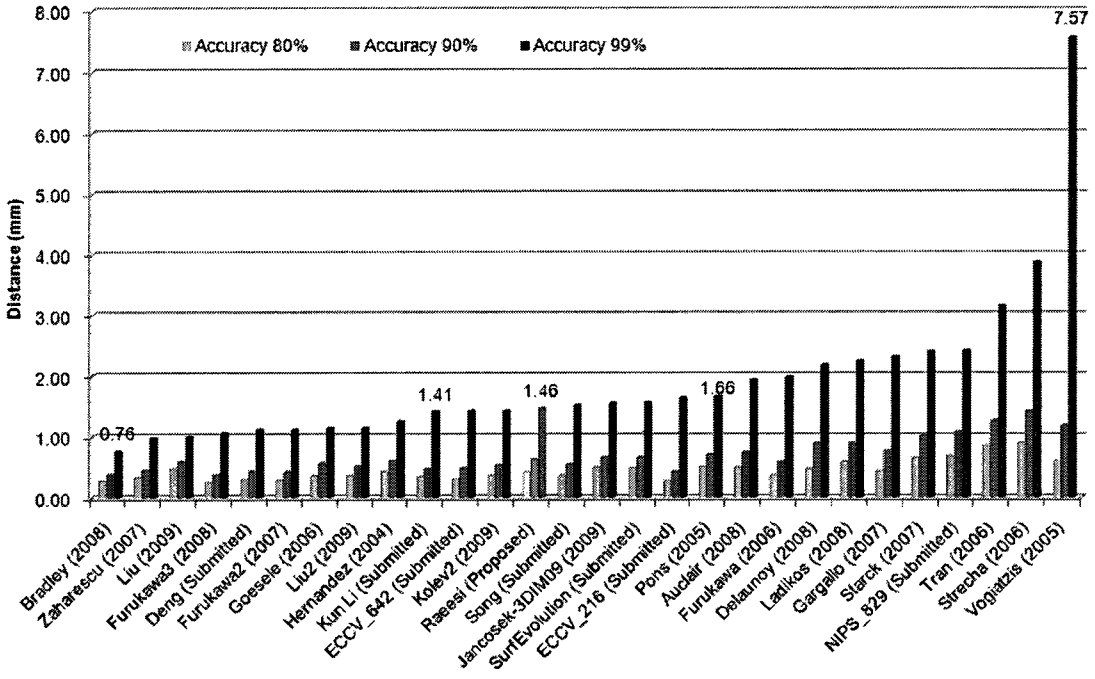


Figure B-1. Accuracy comparison among all the state-of-the-art methods.

The comparison of completeness metric has been shown in Figure B-2 for two different distance thresholds, 1.25*mm* and 1.5*mm*. The completeness values are so close to each other as well. The methods in this plot are sorted for completeness for distance threshold 1.5*mm*.

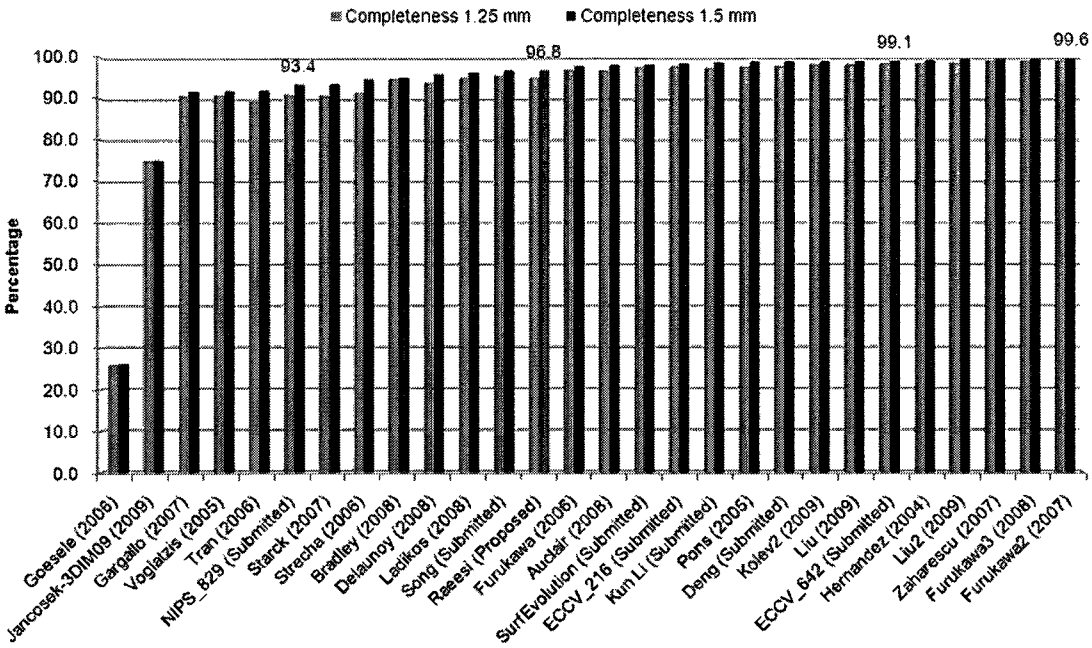


Figure B-2. Completeness comparison among all the state-of-the-art methods.

Figure B-03 and Figure B-4 show the comparison of accuracy and completeness metrics among the existing and submitting depth map based methods for different thresholds.

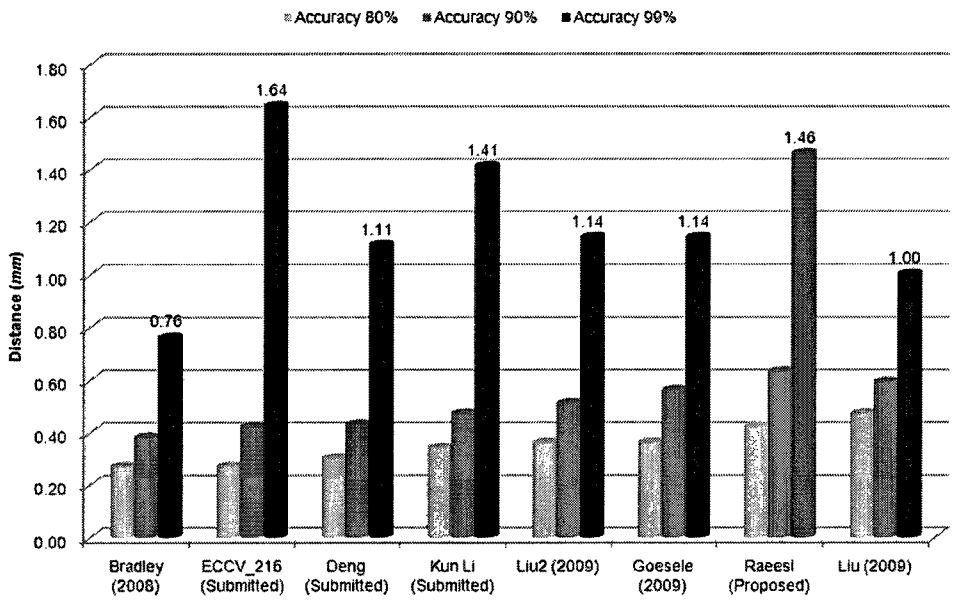


Figure B-03. Accuracy comparison among the depth map based methods.

As it can be seen clearly in Figure B-03, the ranking of the accuracy for threshold 99% is completely different from the ranking for threshold 80%.

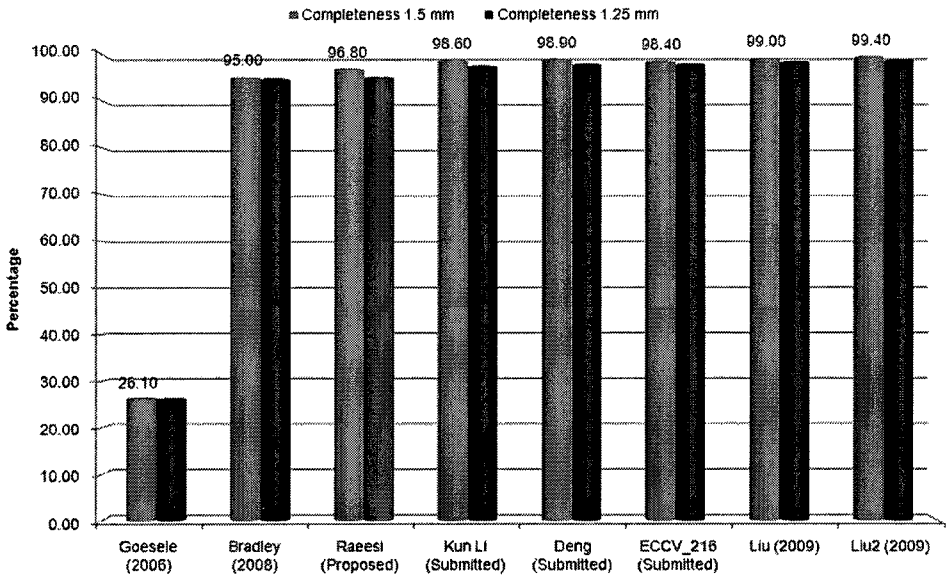


Figure B-4. Completeness comparison among the depth map based methods.

The comparison of the proposed model with the methods which are recently submitted and not published yet are presented in Figure B-5 and Figure B-6.

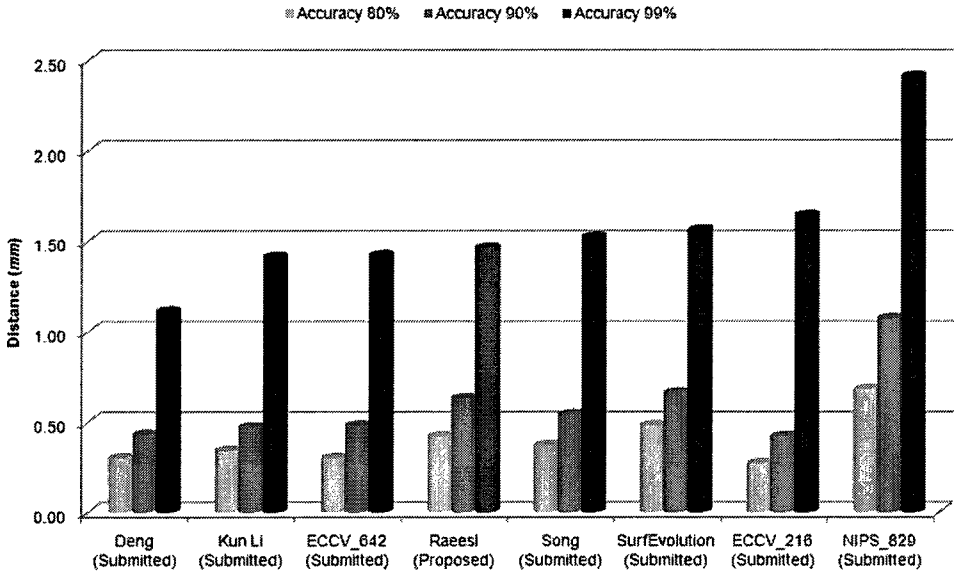


Figure B-5. Accuracy comparison among the methods which are not published yet.

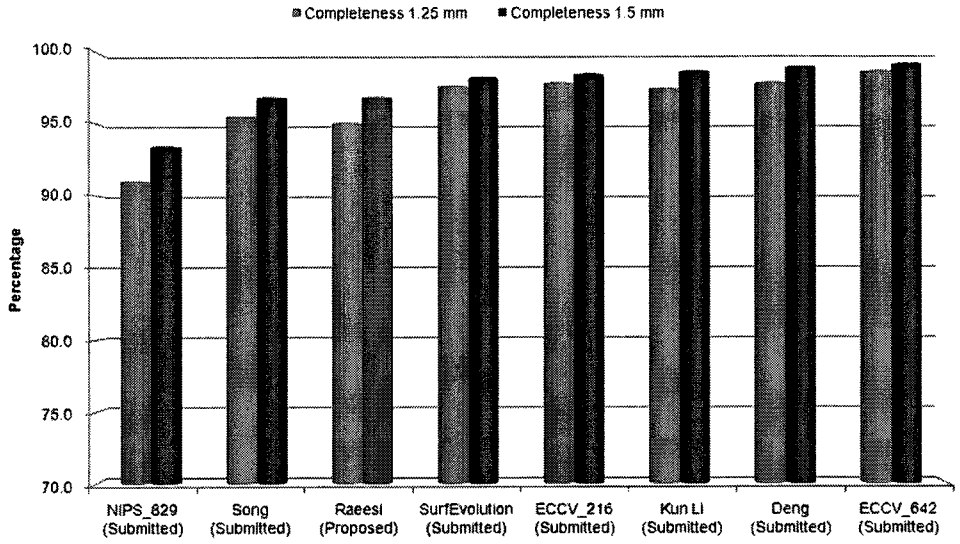


Figure B-6. Completeness comparison among the methods which are not published yet.

VITA AUCTORIS

NAME:	Mohammad R. Raeesi N.
PLACE OF BIRTH:	Esfahan, IRAN
YEAR OF BIRTH:	1985
EDUCATION:	NODET (National Organization for Development of Exceptional Talents) High School, Esfahan, IRAN 1989-2002
	NODET (National Organization for Development of Exceptional Talents) Pre-University, Esfahan, IRAN 2002-2003
	Sharif University of Technology, Tehran, IRAN 2003-2007 B.Sc.