國 立 交 通 大 學

電 機 與 控 制 工 程 學 系

碩 士 論 文

應用於無背景限制下三維建模之物體萃取演算法

A Novel Object Extraction Algorithm in 3D Model Reconstruction for Objects in Arbitrary Background

研 究 生：李 士 昌

指導教授：陳 永 平　博士

中 華 民 國 九 十 六 年 六 月

應用於無背景限制下三維建模之物體萃取演算法

A Novel Object Extraction Algorithm in 3D Model Reconstruction for Objects in Arbitrary Background

研 究 生：李士昌　　　Student：Shih-Chang Li

指導教授：陳永平　　　Advisor：Yon-Ping Chen

國 立 交 通 大 學

電 機 與 控 制 工 程 學 系

碩 士 論 文

A Thesis

Submitted to Department of Electrical and Control Engineering

College of Electrical and Computer Engineering

National Chiao Tung University

in Partial Fulfillment of the Requirements

for the Degree of Master

in

Electrical and Control Engineering
June 2007
Hsinchu, Taiwan, Republic of China

中華民國九十六年六月

應用於無背景限制下三維建模之物體萃取演算法

學生：李士昌　　　　　　　　　　　　　　　　　　　指導教授：陳永平

國立交通大學

電機與控制工程學系

摘要

本論文提出一個可將任意背景影片中的前景物體萃取出來的演算法。由於這個演算法是針對三維物體模型重建系統所設計，因此利用重建系統中常用到的三維資訊重建演算法(camera tracker)做為前置處理器，先行重建出影片中的特徵點及相機位置等的三維資訊。萃取演算法首先將特徵點分為前景及後景兩類，再利用前景的特徵點產生前景物體的遮罩，利用遮罩即可將前景的物體萃取出來。基於這個萃取演算法，本論文同時提出一個三維物體模型重建的原型系統，用以測試萃取演算法之效能。與現存系統的最大差別在於，本系統原型可以處理任意背景的物體影片。由實驗結果得知，雖存在許多需要改進之處，但可驗證本論文提出之演算法及系統原型架構具有高度的可行性，亦可預期在進一步改良之後，本系統及演算法能發揮出之潛力。

A Novel Object Extraction Algorithm in 3D Model Reconstruction for Objects in Arbitrary Background

Student: Shih-Chang Li Advisor: Prof. Yon-Ping Chen

Department of Electrical and Control Engineering

National Chiao Tung University

Abstract

This thesis proposed a novel algorithm for extracting foreground object in an image sequence of arbitrary background. The algorithm is mainly designed for the 3D object model reconstruction system. Hence, a camera tracker is adopted as a pre-processor to obtain the 3D structure information including feature point positions and camera poses of the image sequence. The proposed algorithm first separate the feature points into foreground and background, and then generates a mask from the foreground points to extract the object. Based on the proposed algorithm, a prototype of 3D object model reconstruction system is presented to verify the performance. The proposed system if capable of dealing with object image sequences of arbitrary background, which is not possible for the recent reconstruction systems. The experiment results show that, though it requires improvements in many aspects, the proposed algorithm and system prototype is functional in practice and should become more powerful after the improvements are done.

Acknowledgment

想當初，懵懵懂懂地上了幼稚園，一知半解地上了小學，理所當然地上了國中，從善如流地上了高中，夙夜匪懈地進了大學，最後不可免俗地念了研究所。首先可能要感謝台灣的高等教育制度，給我非寫論文不可的動力，讓我得以在立功、立德、立言中先有了點小小的成就，也對十幾年的學生生涯畫上一個具體的休止符。

這份論文得以順利完成，首要感謝指導教授陳永平老師的悉心指導，在研究過程中給予我許多建議和指正，加深了我思考及探討問題的深度；老師敬業樂群的態度以及對學生的親切關懷，也在待人處事方面給了我最好的身教。同時也感謝口試委員們提出了許多寶貴的意見及建議，讓這份論文能夠更加完善。

而我最要感謝的是我的家人，雖然我待在新竹的時間是待在家裡的三四倍，但因為有這兩三個月的休息充電，讓我有面對學校各種挑戰的動力；也因為有你們的支持、關懷及包容，我才可以無憂無慮地享受這一段求學生活，在大學及研究所裡盡情地揮灑。

最後但是不可或缺的，是這段時間結識的各位朋友們，包括電控系所的學長姐學弟妹和同學、參與 Open House 及學聯會認識的大家、以及族繁不及備載的各位。正所謂「在家靠父母，出外靠朋友」，無論是歡樂的小團體聚餐唱歌八卦嘴炮談心，還是同甘共苦的各項活動籌辦過程，抑或是其他因緣際會下的交集，我會努力記住這些感動。少了你們，這六年就不會這麼多采多姿，也不會有這麼多的照片以及回憶。

如果還要謝下去的話，那還要感謝過去六年中所有我修過課的教授、讓交大順利運作的諸位行政人員及納稅人、餵飽我們肚子的同時也餵飽自己荷包的餐廳和宵夜店、還有發明網路、MSN、BBS 的偉人們。

總之，要謝的人太多了，就謝天吧！

Table of Contents

Abstract (in Chinese) i
Abstract (in English) ii
Acknowledgment iii
Table of Contents iv
List of Tables vi
List of Figures vii

Chapter 1 Introduction 1
1.1 Motivation 1
1.2 Related Works 2
1.3 The proposed system 6
1.4 Organization 7
Chapter 2 Model Reconstruction Using Octree Algorithm 8
2.1 Introduction 8
2.2 Octree 8
2.3 Construction of 3D Octree 11
2.4 Model Triangulation 13
Chapter 3 Object Extraction Using 3D Feature Points 16
3.1 Existing Image Segmentation Algorithms 16
3.1.1 Graphical Partitioning Active Contours 16
3.1.2 Image Segmentation 18
3.2 3D Foreground / Background Separation 20
3.2.1 Problem Analysis on Separation 20
3.2.2 Proposed Foreground Extraction Algorithm 22
3.3 Object Mask Generation 25
3.3.1 Problem Analysis on Mask Generation 25
3.3.2 Object Mask Generation Utilizing the Convex Hull Algorithm 29
Chapter 4 Experiment Results 34
4.1 Octree Reconstruction Result 34
4.2 Background Removal Result 37
4.2.1 The Toy-on-Table Sequence 37
4.2.2 The Statue Sequence 39

4.3 3D Model Reconstruction System Result........41
Chapter 5 Conclusion........44
5.1 Conclusion........44
5.2 Future Work........44
5.2.1 The Camera Tracker........44
5.2.2 The Background Removal Algorithm........46
5.2.3 The Texture Mapping Block........46
Bibliography........47

List of Tables

Table 3.2.1 Foreground Extraction Algorithm 24
Table 3.3.1 Object mask generation algorithm 32

List of Figures

Figure 1.2.1 Camera reconstruction system structure 3
Figure 1.2.2 System structure for depth-map recovery algorithm 4
Figure 1.2.3 System structure for shape-from-silhouette algorithm 5
Figure 1.3.1 Structure of proposed camera reconstruction system 6
Figure 2.2.1 2D Octree diagram 9
Figure 2.2.2 2D Octree structure of Figure 2.2.1 9
Figure 2.2.3 Fail condition for octree 10
Figure 2.2.4 2D octree diagram of different maximum depth R 11
Figure 2.4.1 Reconstructing of sphere using 3D Octree of max depth 3 13
Figure 2.4.2 15 patterns of transforming cube into triangles in Marching Cubes 14
Figure 2.4.3 indexing scheme of Marching Cubes 15
Figure 2.4.4 Triangulation example 15
Figure 3.1.1 GPAC segmentation results 17
Figure 3.1.2 GPAC segmentation results 18
Figure 3.1.3 Image segmentation results 19
Figure 3.2.1 Camera motion when capturing image sequence 20
Figure 3.2.2 Top view of the distribution of reconstructed 3D feature point of a frame 21
Figure 3.2.3 Top view of the distribution of reconstructed 3D feature point of image sequence 22
Figure 3.3.1 Object extraction operation 25
Figure 3.3.2 Ambiguity of determination 26
Figure 3.3.3 Convex hull of Figure 3.3.2 (a) 27
Figure 3.3.4 Ambiguity of concave hull 27
Figure 3.3.5 Concave hull of Figure 3.3.2 (a) 28
Figure 3.3.6 Alpha shape illustration 28
Figure 3.3.7 Alpha shape of Figure 3.3.2 (a) 29
Figure 3.3.8 Process of gift wrapping algorithm 30
Figure 3.3.9 Reduced judgment situation 31
Figure 3.3.10 Object extraction operation 31
Figure 4.1.1 Arrangement for taking object silhouette image 35
Figure 4.1.2 Object silhouettes generated by Maya PLE 8.5 35
Figure 4.1.3 Reconstruction result Figure 4.1.2(a). 36

Figure 4.1.4 Reconstructed ON-cube model of input silhouette 36
Figure 4.1.5 Triangulated model of Figure 4.1.4..37
Figure 4.2.1 Frames of the Toy-on-Table sequence 38
Figure 4.2.2 Foreground/background separation result the Toy-on-Table sequence 39
Figure 4.2.3 Frames of the Statue sequence 40
Figure 4.2.4 Foreground/background separation result of the Statue sequence 41
Figure 4.3.1 Reconstruction result of Toy-on-table sequence. 42
Figure 4.3.2 Reconstruction result of Statue sequence...42
Figure 5.2.1 System structure for depth-map recovery algorithm.......................................45

Chapter 1
Introduction

1.1 Motivation

3D model reconstruction of an object in real world is an active researching topic because of its wide usage in many aspects. For example, museums can build 3D models of collections for virtual exhibition. The 3D model of body shape could be useful in customizing personal products like clothes, shoes, or furniture. It could also be an interesting application for reconstructing the 3D model of the player into the computer games.

Up to now, most subjects of the reconstruction process have been well-developed such as camera calibration, camera tracking, object modeling and texture mapping. However, extracting an object appropriately from an image sequence is still a problem to solve. Most reconstruction processes avoid the problem by setting limitations on the reconstruction environments, for example a clean background for easy background removal. The limitation on the reconstruction environments results in a highly accurate model, but, on the other hand, makes the reconstruction system no only expensive but also hard to build and operate. Besides, it could also make the system only capable of reconstructing the object of certain size if the viewing area is limited by the static camera.

Hence, this thesis proposed a reconstruction system prototype with an novel object extraction algorithm dealing with complex background of regular usage and tries to maintain the quality of reconstructed model at the same time, which makes the system more feasible and applicable in practice.

1.2 Related Works

Currently, 3D model reconstruction can be conducted by human, or automatically operated by laser scanner or camera. The automatic reconstruction process contains two steps, the shape-modeling and the texture-mapping, to take care of the silhouette and to construct the surface texture of the 3D model, respectively.

Reconstruction by human is the most common but time-consuming way. Usually, a 3D artist takes weeks to months to build a model by several sketches or photos. Besides, the quality of the constructed model depends on the ability of the artist. It is clear that the uncertainty of quality and the cost of time are the crucial problems of human reconstruction. Hence, automatic reconstruction systems are presented to overcome these problems.

Reconstruction by laser scanners[5, 6, 10, 15, 18, 22, 26, 28] is the one that can create a model with shape exactly same as the object to be reconstructed due to the fact that the shape data is obtained by scanning the entire object in all angles. However, the scanner is too expensive and requires special know-how to operate in restricted environments. In addition, the scanning process is affected by the material of object. The system does not perform well on objects with surfaces absorbing light, such as fur or velvet. Finally, the scanner needs extra instruments such as camera to capture the texture information since it can not obtain texture information while scanning shape data.

Compared with the two methods above, reconstruction using camera not only attains results of good quality, but also it is much cheaper than laser scanners and takes less time than human. This makes algorithms using cameras become the topic that has been widely investigated among all 3D reconstruction systems. Since 2D images captured by camera contain both shape and texture information of a 3D object, it is the most critical issue to find methods which can extract useful data for reconstruction 3D models.

The typical architecture of camera reconstruction systems is shown in Figure 1.2.1, including three main steps, the camera-calibration, the shape-modeling, and the texture-mapping. The first step is the camera calibration to determine essential optical parameters and perspective characteristics. The second step performs the reconstruction of 3D model shape. The last step is the texture-mapping to generate the surface texture of the model. A preprocess of camera calibration is applied to obtain image characteristics of the camera, which is an important information for the precision of shape modeling step

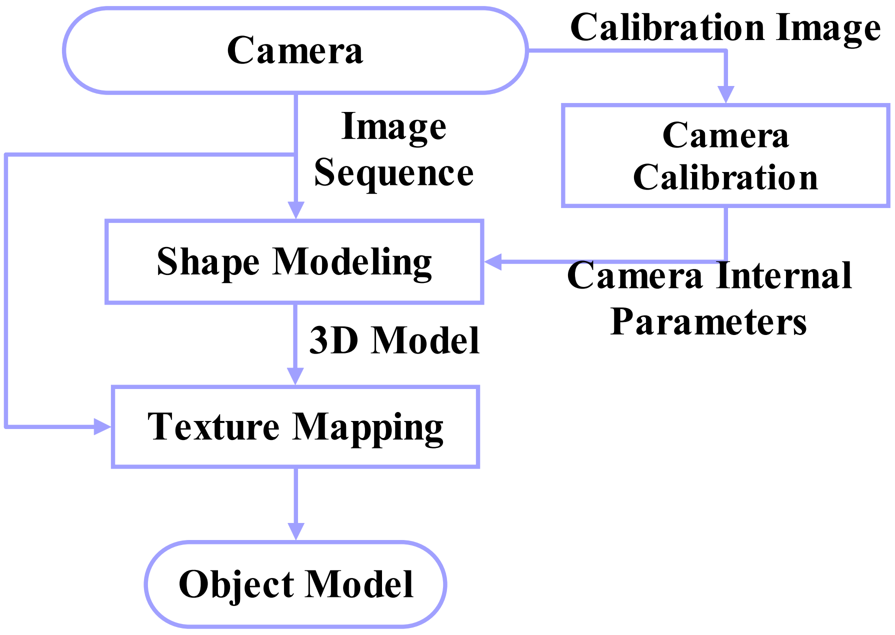

Figure 1.2.1 Camera reconstruction system structure

Most systems [3, 4, 11, 14, 20, 27, 30, 33, 35, 37] use an image sequence, at least 10 images rather than a single image, as input, because single image can not provide sufficient information for the whole process. Generally, the performance of shape-modeling process is affected by the number of input images; in other words, the more input images, more data in other words, the more accurate 3D model.

Systems for 3D model reconstruction nowadays can be classified into two types according to the algorithm used in the model reconstruction block. One type of algorithm is depth-map recovery [3, 11, 20, 27], reconstructs 3D model by estimating the 3D position of every pixel in the input image sequence according to the estimated camera pose. Another type is the shape-from-silhouette algorithm [4, 14, 30, 33, 35, 37], uses the object silhouette in each image to refine the outline of 3D object model according to the corresponding object pose, since the 2D object silhouette is the projection of 3D object shape.

Diagrams of system structure using depth-map recovery algorithm and shape-from-silhouette algorithm are illustrated in Figure 1.2.2 and Figure 1.2.3, respectively. The shape-modeling step is subdivided into several processing block: camera tracking, model reconstruction, and object extraction. The former two blocks are contained in both types of systems, each with different functionality of recover 3D poses of camera and feature points for each image, and reconstruct 3D model from provided data. The object extraction block, which only required by the systems using shape-from-silhouette algorithm, is the most significant difference between these two types of systems.

The differences of system structure and algorithm itself make these two algorithms vary from each other in many aspects. First, the result of depth-map recovery is the 3D model of the whole scene in the image sequence, which is very useful in the whole scene reconstruction but not suitable for reconstruction the object model. On the contrary, the shape-from-silhouette algorithm is designed to reconstruct a specific object in the image sequence. Second, the depth-map recovery algorithm reconstructs the 3D model at pixel-wise precision, but the reconstruction process is very time-consuming. The reconstruction process of the shape-from-silhouette algorithm is much faster than depth-map recovery algorithm because it reconstructs the model to certain accuracy but not exactly. To increase the accuracy, the reconstruction environment must be controlled to obtain errorless information including object silhouettes and object poses. There is another weakness only existing in the shape-from-silhouette algorithm, that is, it can not reconstruct the concave part if the concave never appears in the silhouette.

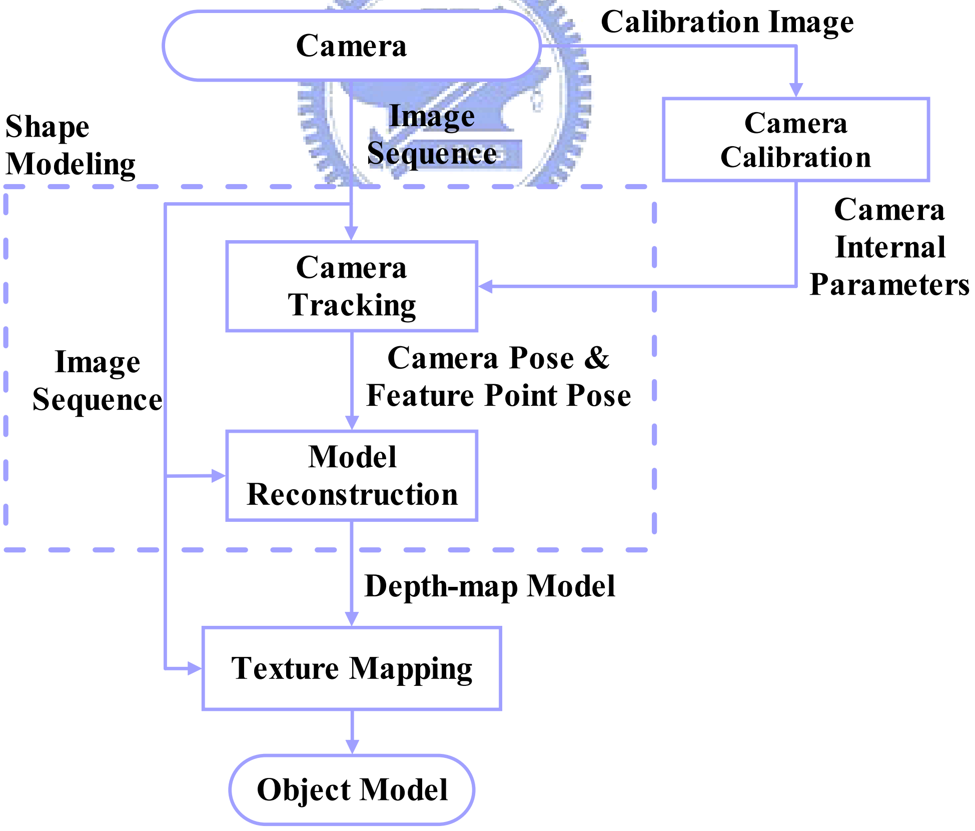

Figure 1.2.2 System structure for depth-map recovery algorithm

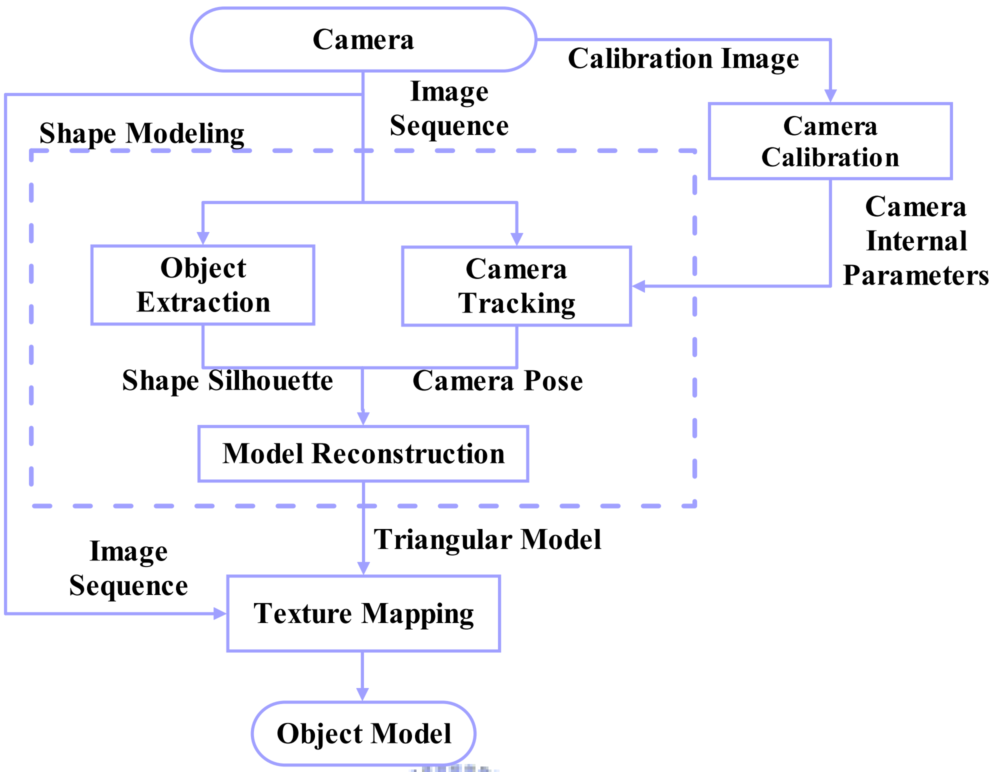

Figure 1.2.3 System structure for shape-from-silhouette algorithm

1.3 The proposed system

This thesis focuses on shape modeling step for the systems using the shape-from-silhouette algorithm and proposes a modified structure which can produce a model of acceptable accuracy without limiting the input image sequence.

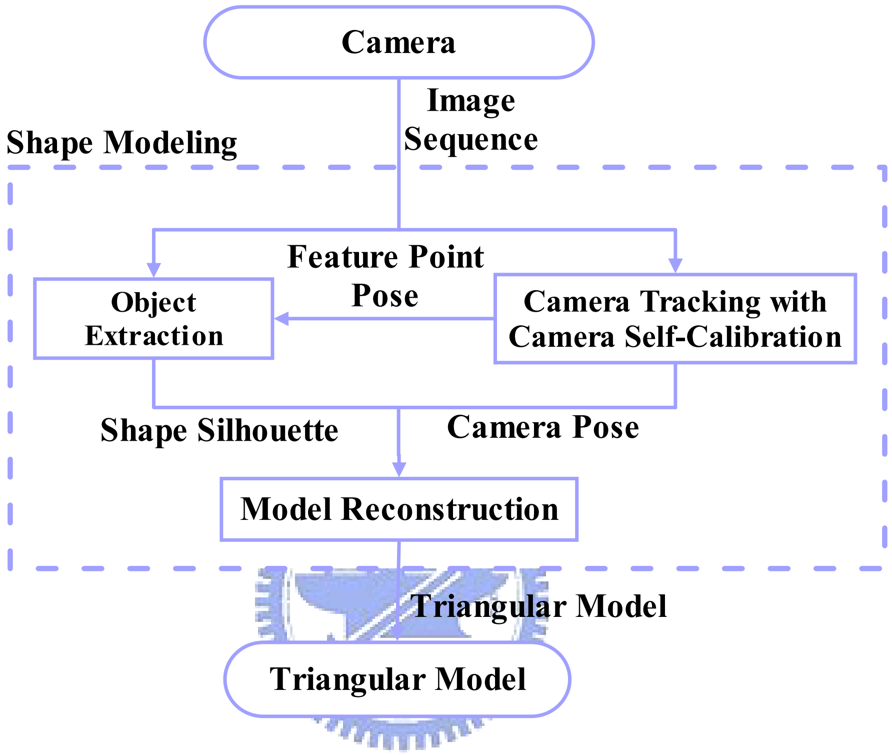

Figure 1.3.1   Structure of proposed camera reconstruction system

Figure 1.3.1 is the proposed camera reconstruction system and the texture mapping step is not included because this thesis only focused on the object shape. The model reconstruction process of proposed system is based on the shape-from-silhouette algorithm. Different from other reconstruction systems, the proposed system applies camera tracking process together with camera self-calibration, a well developed technique, to get the information including camera internal parameter, camera pose, and feature point pose from the image sequence taken freely. Then use the camera tracking results to enforce the object extraction process. After that, the model reconstruction process adopts results from the object extraction process and camera tracking process and performs the shape-from-silhouette algorithm to produce the un-textured object model.

1.4 Organization

This thesis is organized as follows. Chapter 1 gives an overview of the camera reconstruction system and a concept of proposed system. Preliminary techniques used by the proposed system as two system blocks, the camera tracking and the model reconstruction, is introduced in Chapter 2. Next in Chapter 3, the main contribution of this thesis, the back-ground removal using 3D feature points adopted as the object extraction process in the proposed system, is described in detail. The experimental results and conclusions are given in Chapter 4 and Chapter 5, respectively.

Chapter 2
Model Reconstruction Using Octree Algorithm

Octree with marching cube is the most popular 3D model reconstruction algorithm for computer vision or medical image processing. A detailed description of the algorithm is given in Chapter 2.

2.1 Introduction

As mentioned in Section 1.3, the proposed system is based on the shape-from-silhouette algorithm. Among all systems using the same algorithm, the most used model reconstruction method is the octree algorithm [33]. An octree is a hierarchical tree structure consisting of cubes of various sizes in proportion. The use of an octree to represent a 3D model results in several advantages due to the geometry characteristics of cubes.

First, the cube structure is easy to divide in a sequential way, and is also convenient for programming and calculation. Second, any 3D model can be approximately represented by a set of cubes of various sizes. Last but not least, the octree structure is easy to transform into the triangular-mesh model using the "Marching Cubes" algorithm [23, 24] after the octree is built, for the fact that the triangular-mesh model is much smoother and more approximate to the original object because every mesh in 3D space can be separated into triangles.

2.2 Octree

A 2D octree is introduced for easy explanation of the octree structure. Instead of cubes, a 2D octree can be used to represent a 2D diagram composed of rectangles of various sizes, as given in Figure 2.2.1, where a large rectangle could be further divided into four small rectangles. For example, the top rectangle 'A' in Figure 2.2.1 is divided into four rectangles 'A', 'B', 'E', and 'F'. Correspondingly, a 2D octree structure is depicted by a tree of depth 3 in Figure 2.2.2, where each rectangle is called a node. For a node of depth *r* and index *i*, it is denoted as s_i^r with *r* numbered top-down and *i* indexed sequentially. For example, the top node 'A' denoted in gray of depth 1 is denoted as s_A^1 and the bottom node 'D' of depth 3 is denoted as s_D^3.

A node s_i^r represents a rectangle specified by its vertices $v_{im}^r = (x_{im}^r, y_{im}^r)$, $m = 1,2,\ldots,4$, and is labeled as IN, ON, or OUT to denote whether the rectangle is in, on, or out the object contour. If a rectangle is ON, i.e., on the contour, then further partition it into four rectangles, otherwise, leave it unchanged. Viewing from Figure 2.2.1 with a curve as the object contour, the gray rectangle s_A^1 is ON because it contains the object contour, and then further divide it into four green rectangles s_A^2, s_C^2, s_I^2, and s_N^2 of depth 2, where s_A^2, s_C^2, s_I^2 are also ON. In a similar way, s_A^2, s_C^2, s_I^2 can be divided into red rectangles s_A^3, s_B^3, … , s_M^3 of depth 3. As for s_N^2, no division is executed since it is not ON. Once again, it is clear that rectangles s_C^3, s_D^3, s_F^3, s_G^3, s_I^3, s_J^3, s_L^3, s_M^3 are ON and can be further divided in a sequential way.

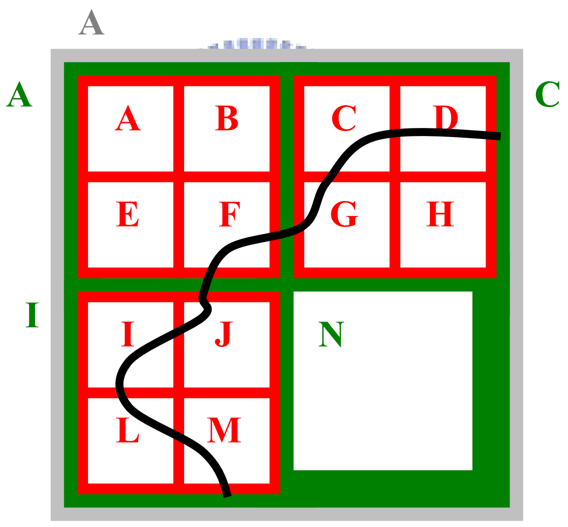

Figure 2.2.1 2D Octree diagram

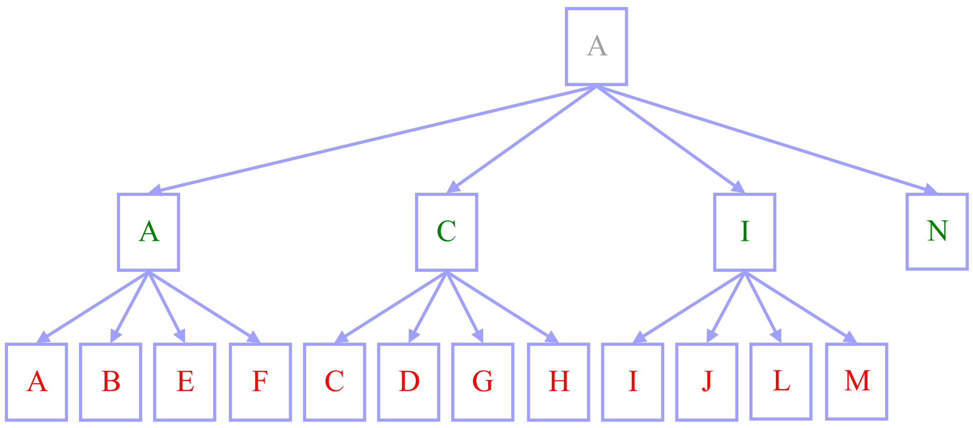

Figure 2.2.2 2D Octree structure of Figure 2.2.1

The formal approach to determine whether the rectangle s_i^r is IN, ON, or OUT is to find one of the intersections of rectangle edges and the object contour. The rectangle is ON if any intersection is detected. However, the formal approach is time-consuming in detecting the intersection by checking all points on the edges. A more efficient detecting way is instead to check only the vertices v_{im}^r, $m = 1,2,...,4$. The rectangle s_i^r can be labeled as IN or OUT for all vertices lay inside or outside the object contour, or as ON otherwise. Since the efficient way only checks the vertices, it fails for the condition as shown in Figure 2.2.3, where the contour is contained inside the rectangle but none of the vertices are inside the contour. The failure will result in defect of octree to decrease the similarity between octree and object contour, and the defect becomes serious for smaller depth *r*. A solution to the condition is to further check $2^{R-r} - 1$ checking points, which are equally distributed on the rectangle edges, and apply the above efficient detecting way to all the vertices and checking points. However, the fail condition still can not be solved on rectangles with depth *R*, unless the formal approach is adopted. But with large R, the effect is relatively small.

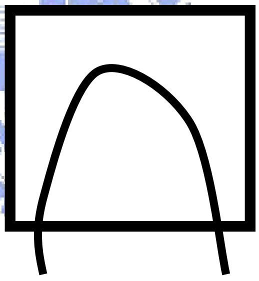

Figure 2.2.3 Fail condition for octree

The accuracy of the octree model depends on the maximum depth *R*, so *R* must be chosen before creating an octree. From the previous example, it is obvious that during the octree construction process, the rectangles of ON will be divided until they are at depth *R*, while the IN and OUT rectangles are kept as large as possible.

To demonstrate the effect related to the maximum depth *R*, another example is given in Figure 2.2.4 for *R* from 4 to 7, where the white region represents the original object shape. Besides, the blue cubes indicate the IN cubes and the green cubes indicate the ON cubes. It is obvious that with the increase of *R*, the shape of octree diagram is approximate to the original object shape further.

Extending the way to a body in 3D space, a 3D octree is obtained to possess cubes as its nodes, denoted as s_i^r of depth *r* and index *i*. Hence, a 3D octree can be represented as

$$T_{3D} = \{s_1^1\} \cup \{s_i^r \mid r = 2,\ldots,R \text{ and } i = 1,2,\ldots,8\}$$

Note that each cube s_i^r has 8 vertices denoted as $v_{im}^r = (x_{im}^r, y_{im}^r, z_{im}^r)$, $m = 1,2,\ldots,8$.

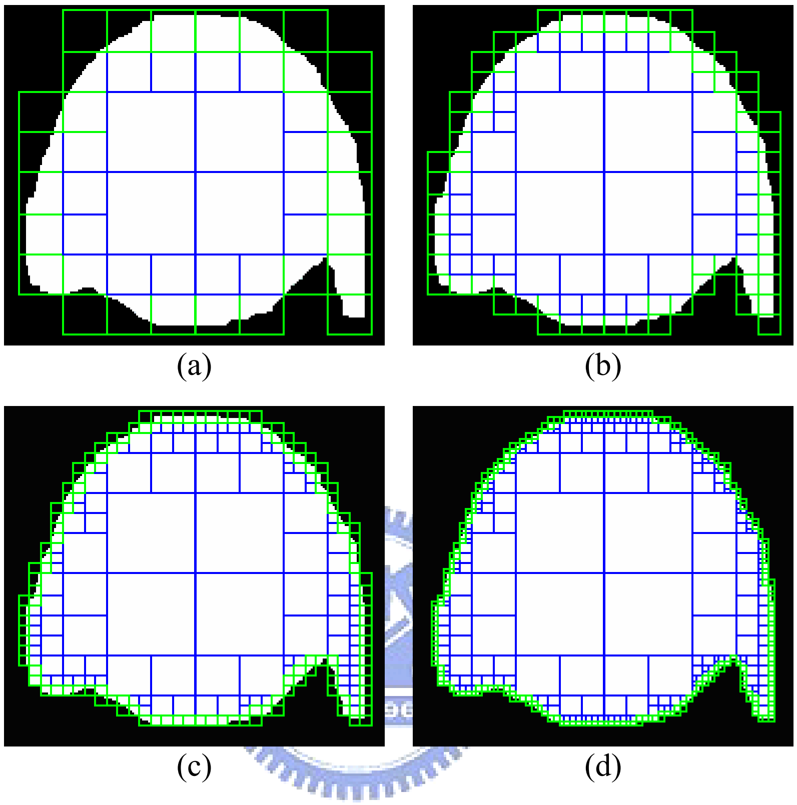

Figure 2.2.4 2D octree diagram of different maximum depth R. Green and blue rectangles representing the ON and IN rectangles respectively : (a) $R = 4$, (b) $R = 5$, (c) $R = 6$, (d) $R = 7$

2.3 Construction of 3D Octree

The first step of 3D object reconstruction is to construct its 3D octree diagram $\mathbf{\Gamma}$. With the estimated object pose denoted as $\{\Delta, \Theta\}$ in the previous section, where $\Delta = (\delta_x, \delta_y, \delta_z)$ and $\Theta = (\theta_x, \theta_y, \theta_z)$ represent the translation and the rotation of the object with respect to the camera in 3D space, an octree node s_i^r with vertices $v_{im}^r = (x_{im}^r, y_{im}^r, z_{im}^r)$, $m = 1,2,\ldots,8$, is then transformed to the vertices $v_{im}^r{}' = (x_{im}^r{}', y_{im}^r{}', z_{im}^r{}')$ as

$$\begin{bmatrix} x_{im}^{r}\,' \\ y_{im}^{r}\,' \\ z_{im}^{r}\,' \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{R}(\Theta) & \boldsymbol{T}(\Delta) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{im}^{r} \\ y_{im}^{r} \\ z_{im}^{r} \\ 1 \end{bmatrix} \tag{2.3.1}$$

where

$$\boldsymbol{T}(\Delta) = \begin{bmatrix} \delta_x & \delta_y & \delta_z \end{bmatrix}^T$$

$$\boldsymbol{R}(\Theta) = \begin{bmatrix} \boldsymbol{C_z C_y} & -\boldsymbol{S_z C_y} & \boldsymbol{S_y} \\ \boldsymbol{C_z S_y S_x + S_z C_x} & \boldsymbol{C_z C_x - S_z S_y S_x} & -\boldsymbol{C_y S_x} \\ \boldsymbol{S_z S_x - C_z S_y C_x} & \boldsymbol{S_z S_y C_x + C_z S_x} & \boldsymbol{C_y C_x} \end{bmatrix}$$

Here, $\boldsymbol{S}_i$ and $\boldsymbol{C}_i$ respectively denote $sin\,\theta_i$ and $cos\,\theta_i$ a for $i = x, y, z$.

After applying (2.3.1) on all vertices of the nodes of octree diagram **Γ**, the resulting octree diagram **Γ'** will have the same pose of the object in the image.

After the pose transformation, the octree diagram **Γ'** is projected onto the image plane by applying perspective projection as (2.3.2) on each vertex of each node of octree diagram **Γ'**.

$$\begin{cases} {}_{2d}x_{im}^{r} = x_{im}^{r}\,' - \dfrac{D_x}{z_{im}^{r}\,' - D_{eye}} x_{im}^{r}\,' \\ {}_{2d}y_{im}^{r} = y_{im}^{r}\,' - \dfrac{D_y}{z_{im}^{r}\,' - D_{eye}} y_{im}^{r}\,' \end{cases} \tag{2.3.2}$$

where

$$D_x = x_{im}^{r}\,' - x_{cam}$$

$$D_y = \boldsymbol{y}_{im}^{r}\,' - \boldsymbol{y}_{cam}$$

$$D_{\boldsymbol{eye}} = \left(\left(x_{im}^{r}\,' - \boldsymbol{z}_{cam}\right)^2 + \left(\boldsymbol{z}_{im}^{r}\,' - \boldsymbol{z}_{cam}\right)^2 + \left(\boldsymbol{z}_{im}^{r}\,' - \boldsymbol{z}_{cam}\right)^2 \right)^{\frac{1}{2}}$$

$(x_{cam}, y_{cam}, z_{cam})$ represents the position of camera, ${}_{\boldsymbol{2d}}\boldsymbol{x}_{im}^{r}$ and ${}_{\boldsymbol{2d}}\boldsymbol{y}_{im}^{r}$ represent the projected 2D position of 3D point $v_{im}^{r}\,'$.

Same as the 2D octree, then each vertex is checked whether it is lying inside or outside the object mask. The octree node can be adjusted and labeled as IN or OUT for all vertices lay inside or outside the object mask, or as ON otherwise.

When processing a series of images, two addition rules are added to the octree adjustment. First, once a cube is labeled as OUT, the cube would never be adjusted again, since it is clear that it is impossible for a cube lies outside the object in one viewing angle but lies inside the object in another angle. Second, an ON cube may be re-labeled as OUT, but can not be re-labeled as IN, for the reason that 3D octree is reconstructed in 2D image plane, the OUT cube may be projected inside or on the object mask and the ON cube may be project inside the object mask. The first situation conflicts with the first rule and the cube must be labeled as out. On the other hand, the ON cubes in second situation become IN cubes due to the model motion and project, they should be kept as ON cubes.

After applying adjustment to the octree with all input object mask and object pose, an approximated model of the object is obtained, as shown in Figure 4.

2.4 Model Triangulation

After the 3D octree construction, the ON cubes of the octree diagram **O** represent an approximation 3D model of the object and the larger depth *R* is, the more accurate the octree diagram becomes. However, the depth *R* is limited to about 7 due to the constrain of computational resources, for example the memory and computational time increase proportionally to the triple order of the depth R. With limitation on the depth *R*, triangulation of the octree not only increases the accuracy of the model but also makes the texture mapping much easier. Figure 2.4.1 shows the reconstruction result of a sphere using a 3D octree of max depth 3. Though max depth 3 is an extreme case which would not be chosen in the reality reconstruction system, it is a good example showing the difference between octree model and Triangulation model

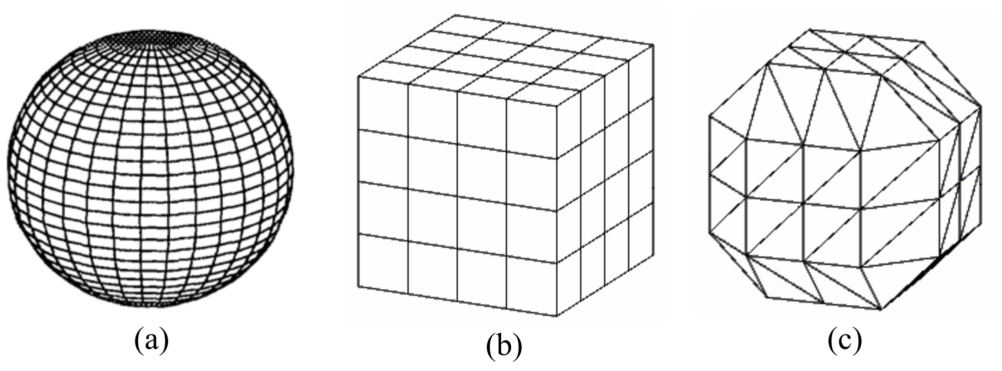

Figure 2.4.1 Reconstructing of sphere using 3D Octree of max depth 3 : (a) original model, (b) octree model, (c) triangulation model of (b)

A typical triangulation algorithm called “Marching Cubes” , which is also adopted in this thesis, is often applied to the cubic-type 3D model reconstruction, such as the octree. The “Marching Cubes” algorithm is based on the fact that the real object surface would intersect with the edges of the ON cubes. By finding the intersection points cube by cube, triangles can be constructed by the geometry relationships of these intersection points. Figure 2.4.2 illustrated 15 patterns of transforming cube into triangles in the Marching Cubes algorithm.

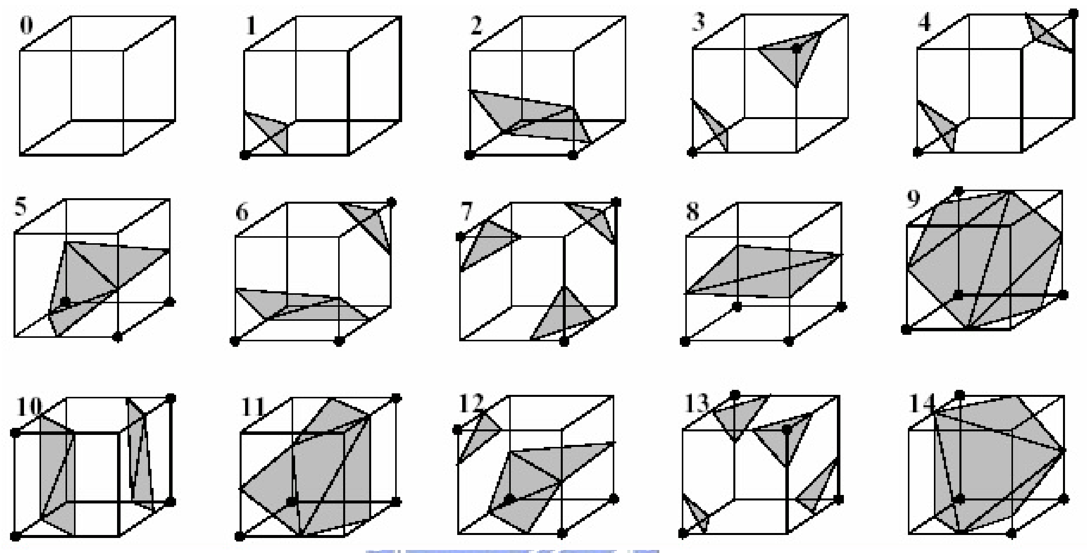

Figure 2.4.2 15 patterns of transforming cube into triangles in Marching Cubes. The dotted corners represent corners lay inside the surface[23]

It is difficult to represent the 256 geometry relationships for a cube in mathematical equation since each vertex of the cube may lie inside or outside the surface. Hence, a general way to implement the Marching Cubes algorithm is to encode the geometry relationships of cube vertices into a unique index number ranging from 0 to 255. The indexing scheme is shown in Figure 2.4.3. The table below the cube shows the correspondence of the eight bits to the vertices v_i, $i = 1...8$, of the cube. Each bit of index number is assigned with 1 if the corresponding vertex lay inside the surface or 0 otherwise. As a result, a unique index number is generated and then used to find out the corresponding triangle structure, which is defined using the edge number e_j, $j = 1...12$, in the look-up table.

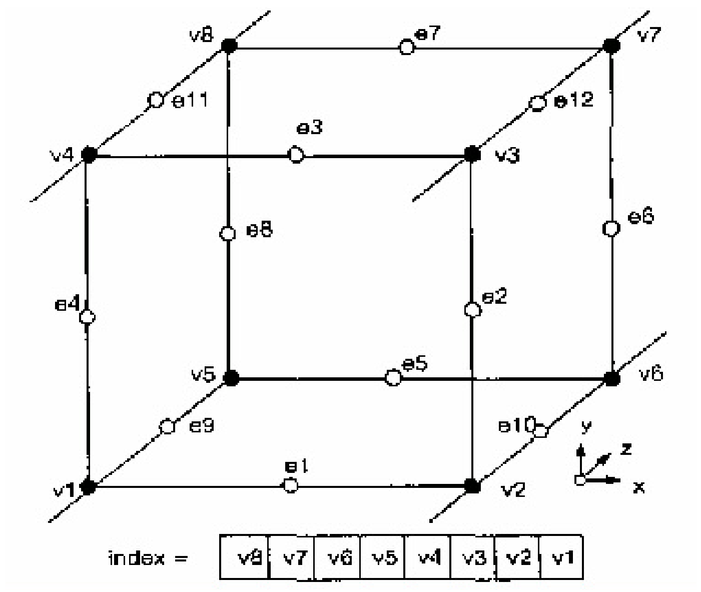

Figure 2.4.3 indexing scheme of Marching Cubes.[23]

From Section 2.2.4, every ON cube should have some vertices laying inside the object and others laying outside, making some edges of the ON cube intersect with the object surface. Thus these edges should have an inside vertex in one end and an outside vertex in the other end, which means an intersection point exists on each of these edges. The exact intersection points on these edges can be determined by geometry algorithms such as binary search [37]. After the intersection points are obtained, the marching cubes algorithm is then adopted to construct the triangle mesh model. Figure 2.4.4 gives an example of triangulation result of marching cubes algorithm.

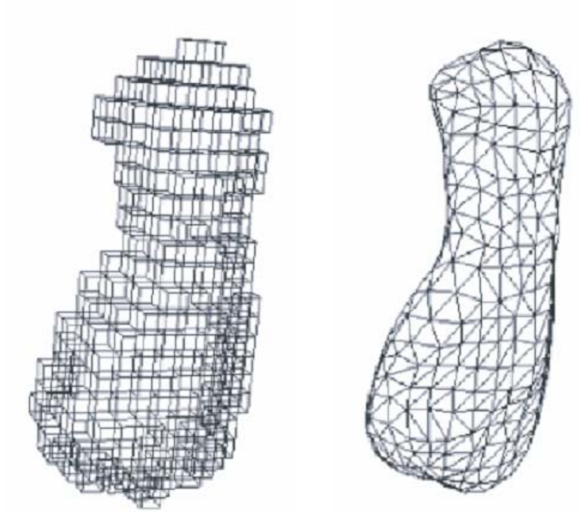
(a) (b)

Figure 2.4.4 Triangulation example[37]: (a) octree model, (b) triangulation model of (a)

Chapter 3
Object Extraction Using 3D Feature Points

Segmentation of foreground and background of an image is crucial many computer vision applications. Two existing and developing algorithms are introduced in Section 3.1. Based on structure of the proposed reconstruction system, a new algorithm is proposed to remove the background and extract the targeting object in the foreground, utilizing the tracking results of camera tracking system mentioned in Section 2.1, including camera poses and the 3D positions of feature points. Two steps of the Algorithms, the 3D foreground / background segmentation and the object mask generation, are explained in the Section 3.2 and 3.3.

3.1 Existing Image Segmentation Algorithms

3.1.1 Graphical Partitioning Active Contours

"Active Contour"[19, 36] is the most used algorithm for object extraction. An active contour, or a snake, can be represented using (3.1.1).

$$v(s) = (x(s), y(s)),\ s \in [0,1] \tag{3.1.1}$$

where s represents the indexing value of each points belong to the active contour, as $v(s)$ represents the position of the point indexed by s. The deformation of an active contour is controlled by an energy function defined by the image information like color and edge, as shown in (3.1.2)[32].

$$E_{snake} = \int \left(E_{internal}(v(s)) + E_{image}(v(s)) + E_{constraint}(v(s)) \right) ds \tag{3.1.2}$$

where $E_{internal}$ represents the energy of smoothness defined by elasticity and stiffness parameter, E_{image} represents the energy defined by image information including color, texture and edge, and $E_{constraint}$ represents the energy defined by information other than the image like object shape pattern, respectively.

The position and shape of active contour is the one that makes E_{snake} minimal, hence an iterative process of minimization is required. Many minimization process with an implementation of active contour have been proposed. The most studied one among these algorithms is the "Level set method"[29]. Instead of deforming the contour, the level set method assigns a height value $c(x,y)$ for each point (x,y) in image and uses the height value to determine the location of contour according to (3.1.3). The height value $c(x,y)$ is iteratively updated by calculating the effect of E_{snake} to point (x,y) until saturation, which indicates no sign changes for any $c(x,y)$. As the iteration process is completed, the contour of the targeting object is obtained.

$$c(x,y)\begin{cases} <0, & \text{outside contour} \\ >0, & \text{inside contour} \\ =0, & \text{on contour} \end{cases} \tag{3.1.3}$$

The GPAC[31] algorithm is proposed recently for the sake of foreground/background segmentation of nature pictures, as shown in Figure 3.1.1(a) and 3.1.1(b). However, the segmentation result of GPAC algorithm shown in Figure 3.1.2(b) indicates a unexpected result in segmenting the test image as Figure 3.1.2(a). Viewing from Figure 3.1.1, the GPAC algorithm is well-performed under the condition that the foreground and background are different in color tones. When the color of foreground object is similar to the background, the segmentation is failed.

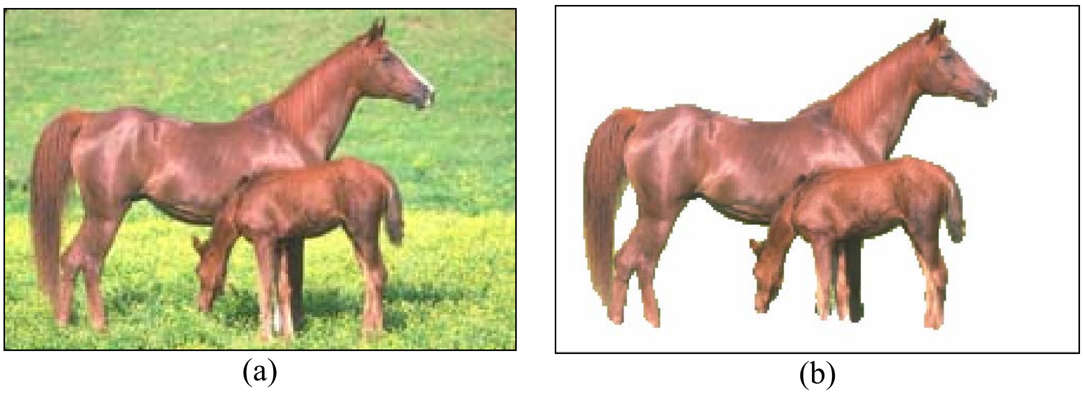

Figure 3.1.1 GPAC segmentation results : (a) test image 1, (b) extracted foreground of (a)

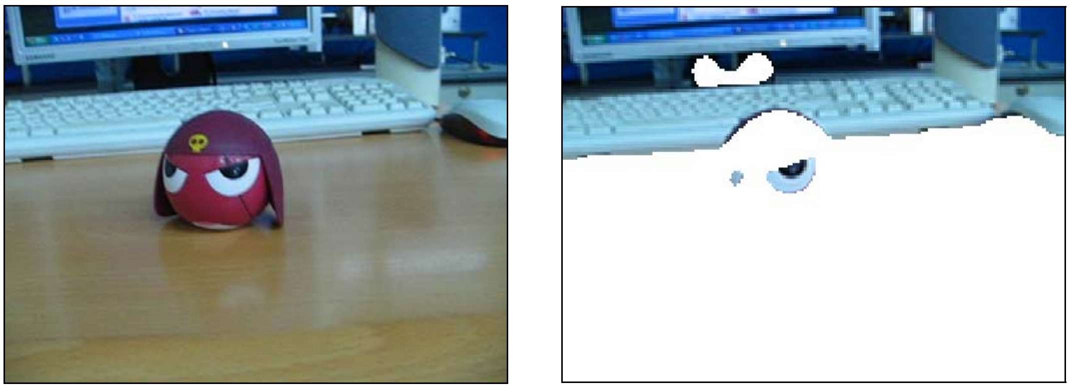
(a) (b)

Figure 3.1.2 GPAC segmentation results : (a) test image 2, (b) extracted foreground of (a)

3.1.2 Image Segmentation

Different from GPAC, image segmentation algorithms tend to divide the image into several segments according the segmentation condition, usually determined by the color and edge information. The objective of image segmentation algorithm is to divide the image into several segments such that each segment contains pixels with similar features and is mostly distinct to other segments. The basic image segmentation algorithm is the watershed algorithm, which simply calculates the boundary of every pixel with local minimum color. The watershed algorithm always over-cut the image and is applied as a pre-process of other segmentation algorithms, as shown in Figure 3.1.3(b). Improvements are applied to watershed algorithm including merging similar segment areas and using more information like texture and edge for calculation. EDISON[8, 16] is a segmentation algorithm implements the improved algorithm, and the segmentation result is shown in Figure 3.1.3(c).

From the segmentation result of EDISON in Figure 3.1.3(c), it is obvious that the over-cut problem still exists to be solved for the algorithms based on watershed. Hence, the graph-based image segmentation algorithm[12, 38] is introduced. Instead of image processing theory, the algorithm segments image based on graph theory. The image to be segmented as a graph $G=(V,E)$, where $v_i \in V$ represents pixels in image and $e_{ij} \in E$ represents connecting edge of two neighboring pixels v_i and v_j. A weight value $w_{ij}=w(v_i,v_j)$ is calculated for every edge e_{ij} according to the dissimilarity of pixel v_i and v_j. With the weight value w_{ij}, the graph-based segmentation algorithm is able to segments the image into several sub-graphs $G_k'=(V,E_k')$ with properties $G_k' \in G$, $E_k' \in E$, and $G_k' \cap G_l' = \Phi$,

$E_k' \cap E_l' = \Phi$ for $k \neq l$, while satisfying the objective of image segmentation algorithm mentioned above. A graph-based segmentation proposed in [12] has been tested and the result is illustrated in Figure 3.1.3(d).

Though the image segmentation algorithm is performs well in divide image into image blocks, it can not determine if the image block belongs to foreground or background. It is impossible to piece image blocks together to form a complete object image since the algorithm provides nothing about the geometry of the object. Therefore, the image segmentation algorithm is not suitable for the reconstruction system, either.

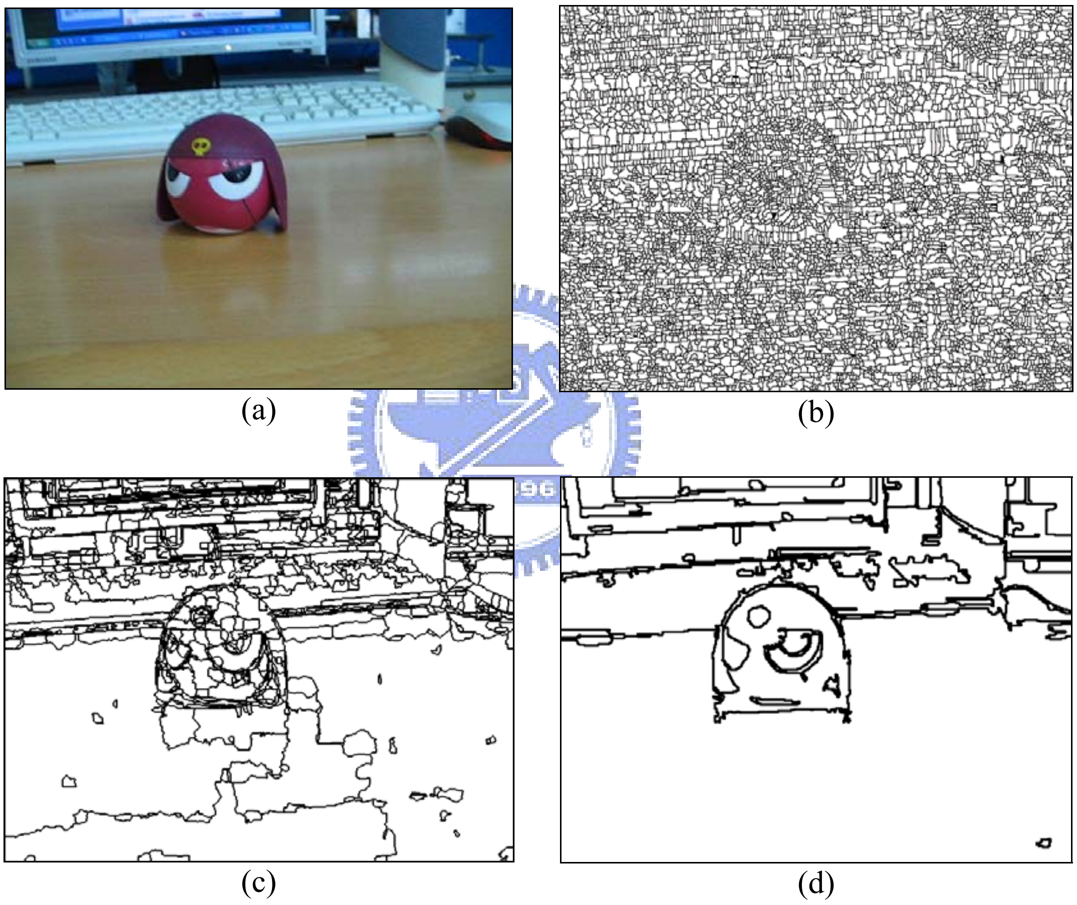

Figure 3.1.3 Image segmentation results : (a) test image, (b) result of watershed algorithm, (c) result of EDISON[16], (d) result of graph-based segmentation algorithm proposed in [12]

3.2 3D Foreground / Background Separation

3.2.1 Problem Analysis on Separation

To obtain an image sequence containing an object suitable for reconstruction, the image sequence must be taken by a camera orbiting around the target object, as shown in Figure 3.2.1. To satisfy the above condition, the shooting environments should be arranged to keep the target object from the surrounding background in a certain distance. To reconstruction the 3D information from the image sequence, the camera tracking system is adopted to reconstruct the camera 3D pose and 3D feature point positions. With reconstructed 3D feature point positions, the separation between the target object and the background will be also restored by the camera tracking system. As a result, the reconstructed feature points are distributed in two groups separated by an empty space gap. The group close to the camera is identified as the foreground, while the other one far away from the camera is identified as the background. The average distance between points of the foreground is much smaller than that of the background, as illustrated in Figure 3.2.2. The objective of separation is to cluster the reconstructed feature points into two groups of foreground and background.

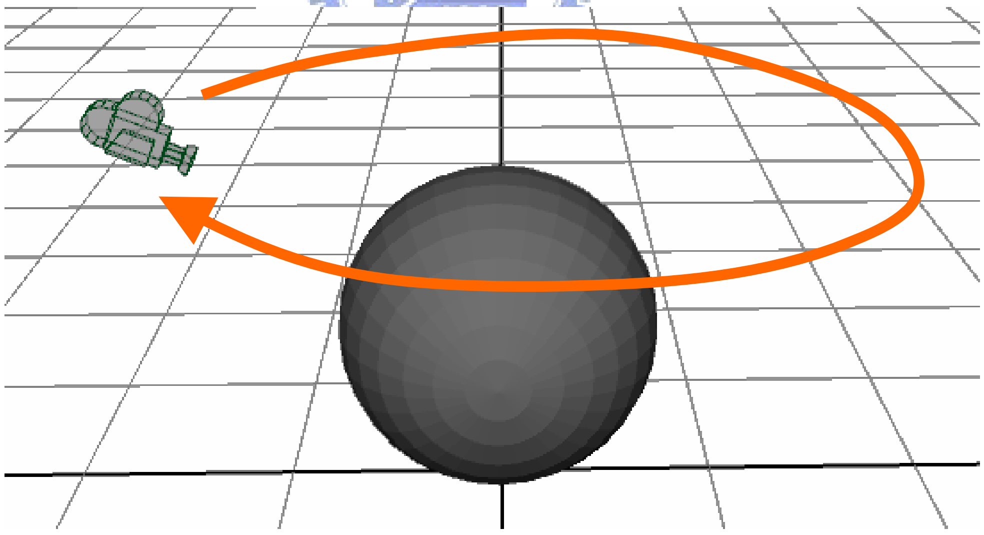

Figure 3.2.1 Camera motion when capturing image sequence. The camera is always aiming at the object. Illustration generated using Maya PLE 8.5[2]

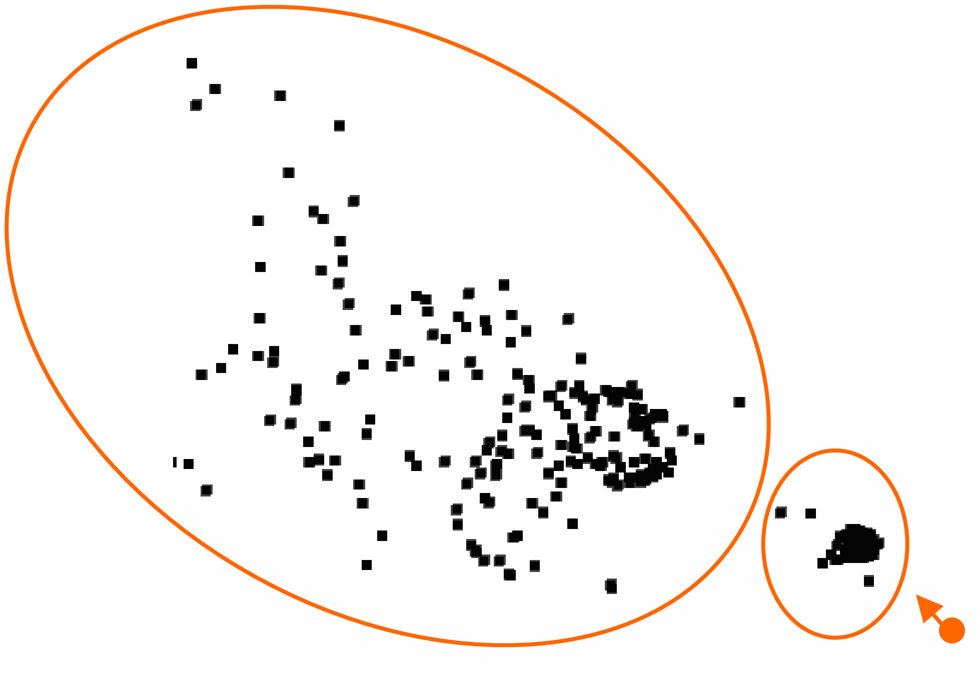

Figure 3.2.2 Top view of the distribution of reconstructed 3D feature point of a frame. Circle and arrow at right bottom represent the position and viewing direction of the camera, respectively. The smaller ellipse indicates the foreground points while the larger ellipse indicates the background points.

Though there is a clear separation between the two point groups, the clustering algorithm like K-means is not applicable for two reasons. First, most clustering algorithm attempts to separate 3D data using planes. However, the boundary shape of foreground and background group is more likely a sphere or an irregular shape, not a plane, as shown in Figure 3.2.3. Second, it is unnecessary to find a separation surface to extremely separate the 3D feature points into two groups of foreground and background. Instead, the background removal can be performed by an equivalent process, the foreground extraction, by picking out the foreground points from the feature points.

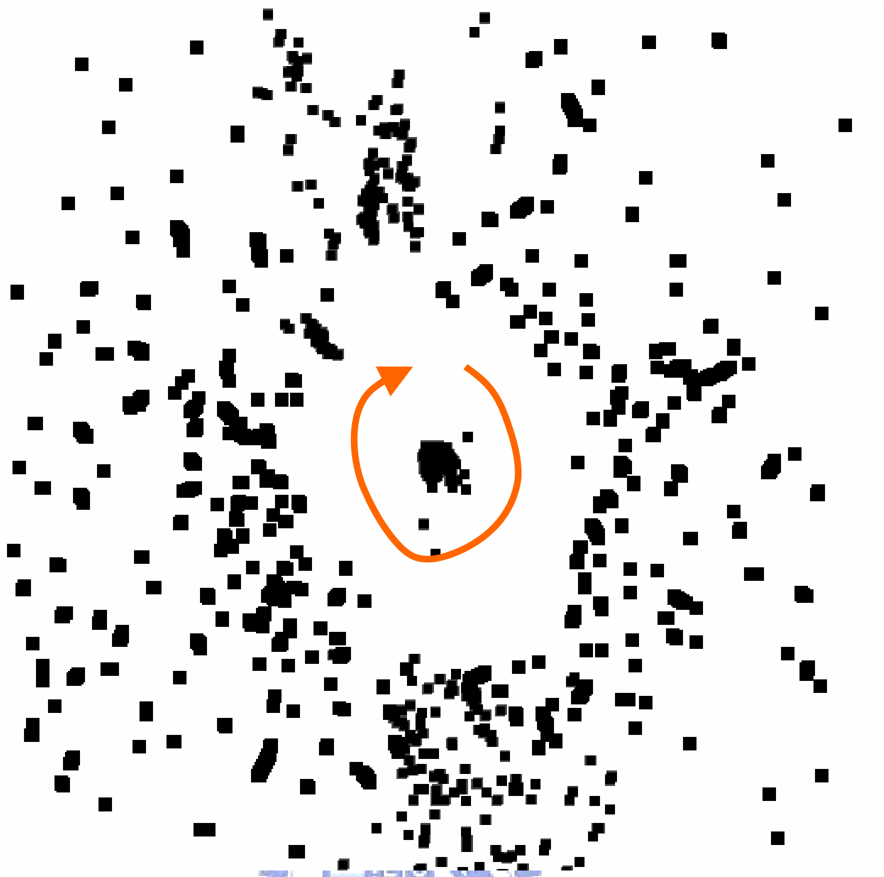

Figure 3.2.3 Top view of the distribution of reconstructed 3D feature point of image sequence. The orange arrow indicates the orbit of camera.

3.2.2 Proposed Foreground Extraction Algorithm

With reconstructed camera position and feature point location information, an algorithm performing foreground points extraction is proposed for this particular condition. Utilizing the distribution characteristic illustrated in Figure 3.2.2, the proposed algorithm first determines the initial two-point set of the foreground group by finding two closest points either or both of them are visible to the camera. The determination was performed by checking the included angle of two vectors, the camera viewing direction and the vector from camera position to the feature point position, by (3.2.1) .

$$\theta_v\left(P_f, P_c, D_c\right) = cos^{-1}\left(\frac{D_f \bullet D_c}{\left|D_f\right| \cdot \left|D_c\right|}\right) \tag{3.2.1}$$

where $P_f = \left(P_{fx},\ P_{fy},\ P_{fz}\right)$, $P_c = \left(P_{cx},\ P_{cy},\ P_{cz}\right)$, $D_c = \left(D_{cx},\ D_{cy},\ D_{cz}\right)$, and $D_f = \left(P_{fx} - P_{cx},\ P_{fy} - P_{cy},\ P_{fz} - P_{cz}\right)$ represents the position of feature point, the camera position, the camera viewing direction, and the direction from camera to feature point, respectively. The point P_f is determined visible to camera if θ_v is smaller than θ_c , usually defined as the viewing angle of the camera, or can be defined as a small angle such as 10 degrees to narrow down the range of foreground object.

A classification process is then performed to iteratively integrate suitable unclassified feature points into the foreground group. An unclassified feature point is selected and integrated if its distance to any foreground group point is smaller than the threshold distance Th_d, defined as (3.2.2).

$$Th_d(W,d,d_1,d_2) = \frac{W}{d \cdot mean(d_1,d_2)} \qquad (3.2.2)$$

where $W = min(ImageWidth, ImageHeight)$, d_1 and d_2 represent the distances from the camera to the two points of the initial point set, d is the expected minimal number of feature points extracted from 2D projection of the object surface.

From (3.2.2), Th_d is proportional to the image resolution since the pixel distance between two feature points becomes larger for higher image resolution. Besides, Th_d is inverse proportional to the distance from camera to the foreground object, due to the fact that the object size in image becomes smaller and the distance between feature points becomes closer when the object is farther. Th_d is also inverse proportional to the expected minimal number of feature points for the reason that the more feature points on an object surface, the closer the feature points.

The detail of proposed foreground extraction algorithm is presented in Table 3.2.1.

Table 3.2.1 Foreground Extraction Algorithm

Input

Minimal dimension of image width and height W, expected minimal number of feature point d, 3D camera position P_c, camera viewing direction D_c, camera viewing angle θ_c, and a set of 3D feature points position $\boldsymbol{P}$.

Output

foreground points set $\boldsymbol{F}$

Algorithm

1. **Initialization** : Determine the initial two-points set of foreground group

```
$d_{min}$ := A_VERY_LARGE_NUMBER
FOR EACH point $P_i$ in $\boldsymbol{P}$
    Find nearest point $P_j$ in $\boldsymbol{P}$ to $P_i$ with distance $d_{ij}$
    IF $\theta_v(P_i, P_c, D_c) < \theta_c$ OR $\theta_v(P_j, P_c, D_c) < \theta_c$ THEN
        IF $d_{ij} < d_{min}$ THEN
            $d_{min} := d_{ij}$
            $\boldsymbol{F}_1 := P_i$
            $\boldsymbol{F}_2 := P_j$
        END IF
    END IF
END FOR
```

2. **Foreground Extraction** : Iteratively integrate unclassified feature points into the foreground group

```
WHILE no more points are added to $\boldsymbol{F}$
    FOR EACH point $P_i$ in $\boldsymbol{F}$
        Find nearest point $P_j$ in $\boldsymbol{P}$ to $P_i$ with distance $d_{ij}$
        IF $d_{ij} < Th_d(W, d, \boldsymbol{F}_1, \boldsymbol{F}_2)$ THEN
            Add $P_j$ into $\boldsymbol{F}$
            Remove $P_j$ from $\boldsymbol{P}$
        END IF
    END FOR
END WHILE
```

3.3 Object Mask Generation

3.3.1 Problem Analysis on Mask Generation

To extract an object in an image, the most used approach is applying an object mask to the image to remove the unnecessary part. Feature points belonging to the object are extracted by the foreground extraction algorithm proposed in Section 3.1, as shown in Figure 3.3.1(b). The next step is to generate a proper object mask from these feature points. For instance, generate the object mask illustrated in Figure 3.3.1(c) from the feature points illustrated in Figure 3.3.1(b).

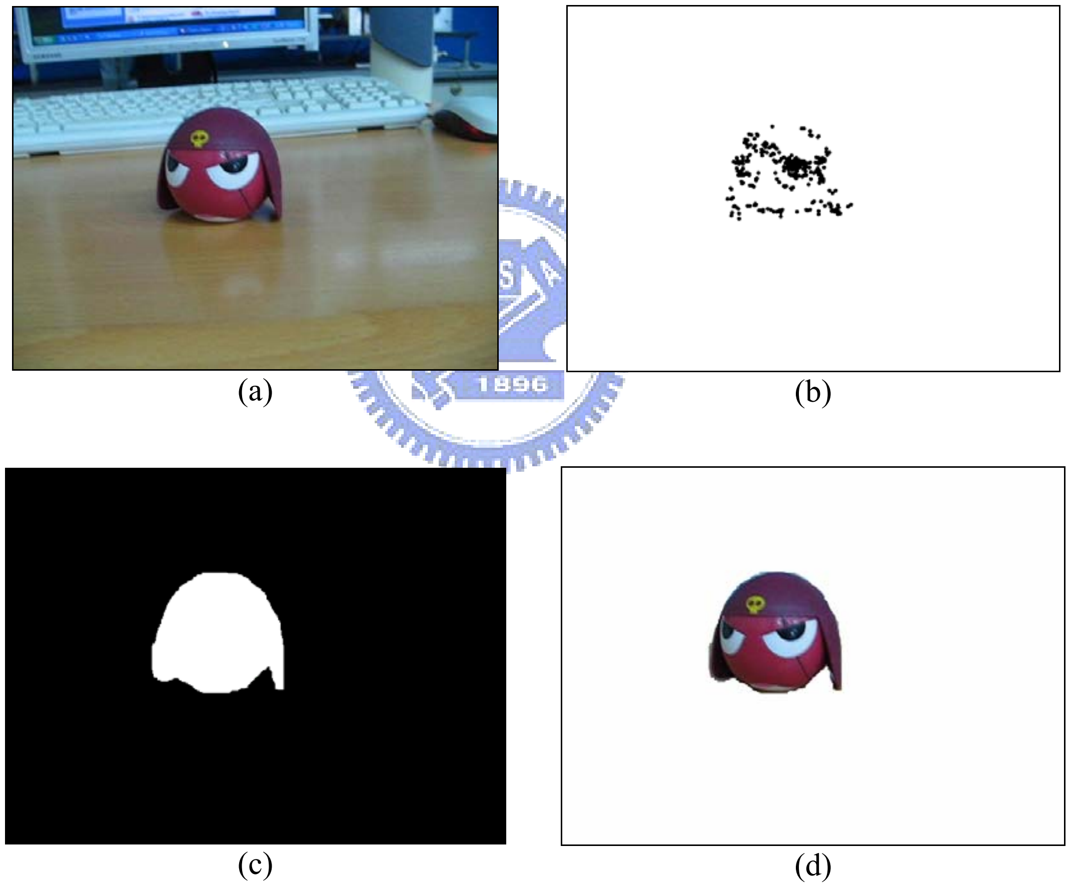

Figure 3.3.1 Object extraction operation : (a) original image, (b) extracted foreground points of (a), (c) ideal generated object mask of (a), (d) ideal extracted object of (a)

A characteristic of the feature points distributed on the edge of object in image can be observed from Figure 3.3.2(a). This characteristic of distribution makes it extremely difficult to generate an object mask from the extracted foreground points in two aspects. First, the distances among feature points are irregular which make the determination of the object contour ambiguous, as shown in Figure 3.3.2(c). Second, the density of feature points varies with respect to the object texture which leads to an ambiguity of distinguishing holes from surface, as shown in Figure 3.3.2(d).

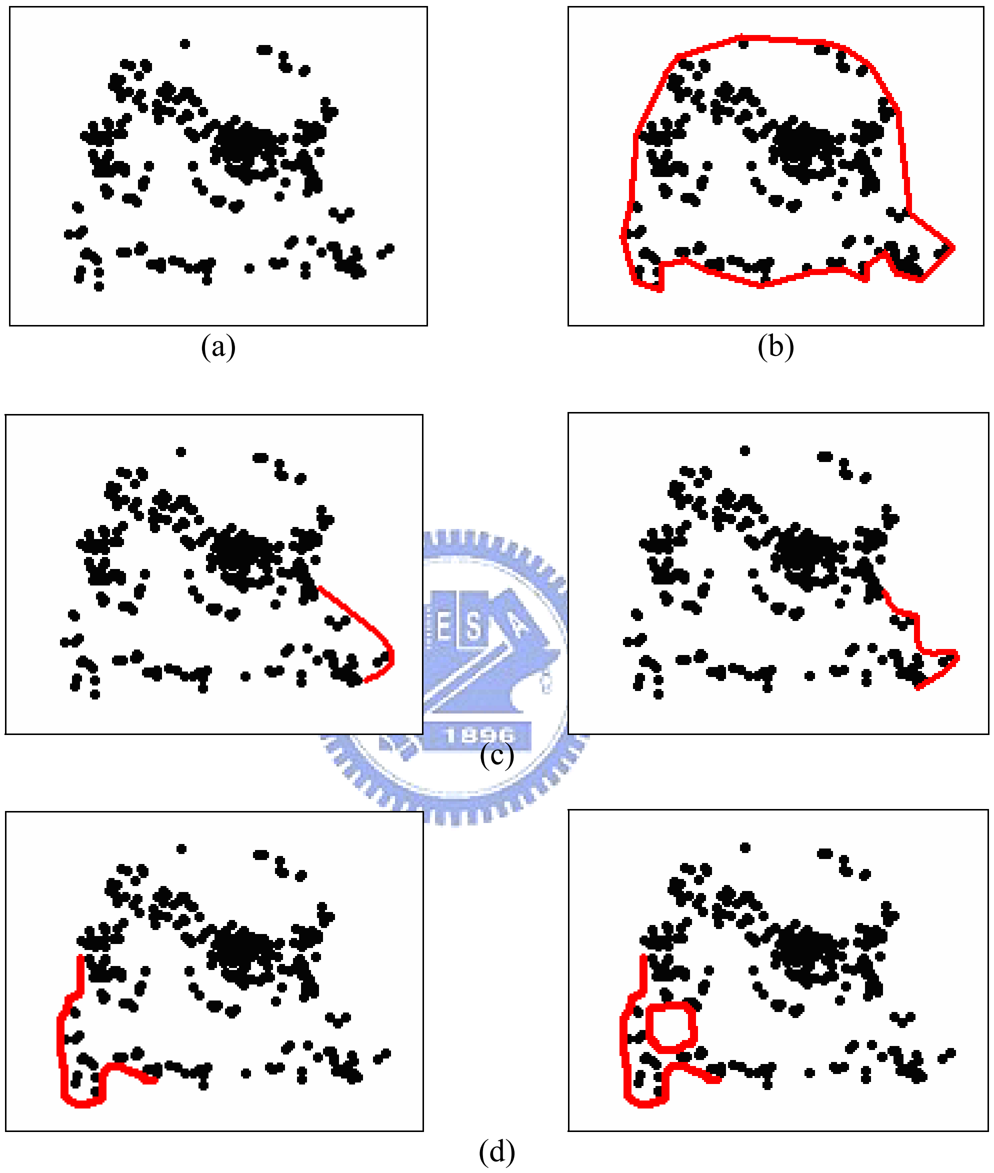

Figure 3.3.2 Ambiguity of determination : (a) the point set, (b) ideal contour of (a), (c) contour ambiguity of (a), (d) hole ambiguity of (a)

Several algorithms have been proposed to deal with the problem of finding best shape fitting a set of points. The most simple and robust algorithm is the convex hull algorithm [9]. The convex hull algorithm generates minimal convex contour containing all points. However, the convex hull algorithm can not deal with holes and the concave parts of the model shape, as shown in Figure 3.3.3.

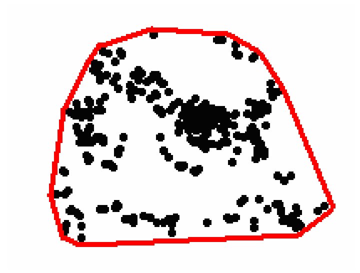

Figure 3.3.3 Convex hull of Figure 3.3.2 (a)

To improve the performance when dealing with concave situation, the concave hull algorithm is proposed based on the convex hull algorithm [25]. Instead of looking for global concave contour containing all points, the convex hull algorithm finds only local concave contour in certain range. Theoretically, some convex parts can be preserved as all points are still contained by the contour. Hence, a new problem of determining the proper contour of the points is introduced since there are various possibilities of choosing the contour path. Generally speaking, the convex hull is smoother with larger range, as shown in Figure 3.3.4. Another issue besides the ambiguity is that, the generated contour is affected by the range of local concave contour and the distribution of the points. Concave hull algorithm performs well only when points are normally distributed. From Figure 3.3.3(a), the density of extracted points from Section 3.2 varies a lot and the result in Figure 3.3.5 illustrated a false contour by the affection of point distribution.

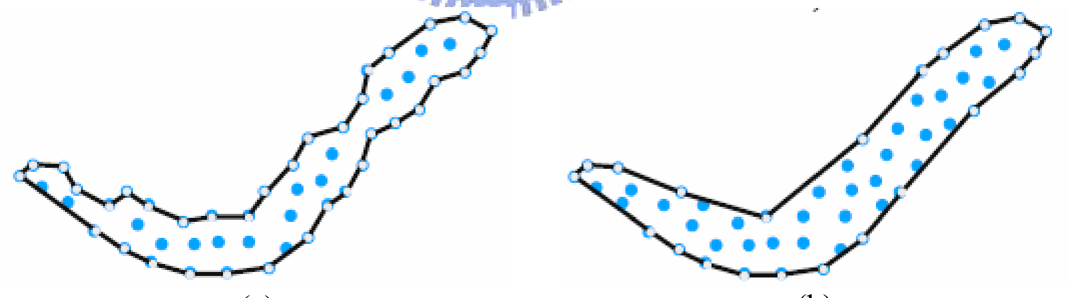

Figure 3.3.4 Ambiguity of concave hull[25]

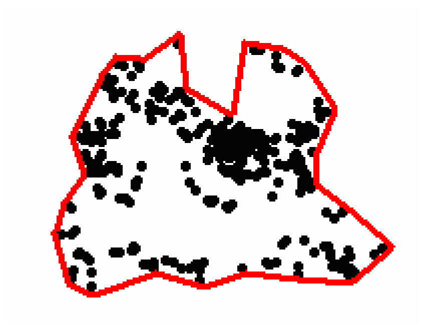
Figure 3.3.5 Concave hull of Figure 3.3.2 (a)

Besides the above two algorithms, alpha shape algorithm [7, 13] is proposed with abilities of not only preserving concave parts but also process holes. Different from using lines to determine the contour, the concept of alpha shape algorithm is to use circles of some radius *r* to determine the contour shape. If no point lies inside some circle, than the circle area is eliminated. After the elimination process, the remaining area represents the shape of the point set, as shown in Figure 3.3.6. Though alpha shape algorithm seems to be capable of dealing with the concerning problem, same problem as concave hull exists. The radius of circle *r* plays an important role in determining the shape: a small *r* results in the misjudgment of holes, while large *r* leads to the neglect of concaves. Observing from Figure 3.3.2 (a), the distribution of points makes is almost impossible to determine a proper *r* for the algorithm. The result of best *r* obtained by a serious try and error is shown in Figure 3.3.7 (a). Although the result in 3.3.7 (a) is very close to the ideal contour shown in Figure 3.3.7 (b), the result is sensitive to *r* which can only obtained by try and error currently.

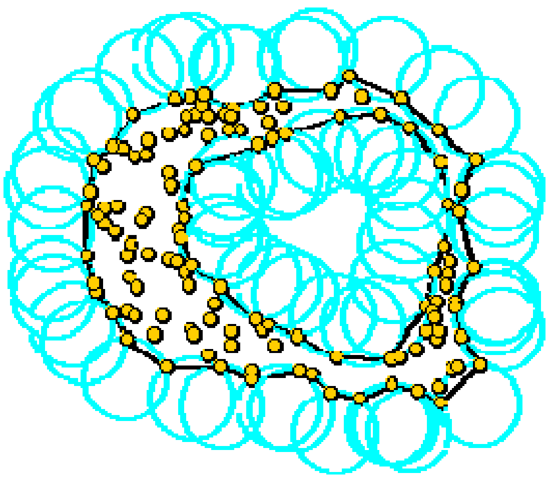
Figure 3.3.6 Alpha shape illustration[7]

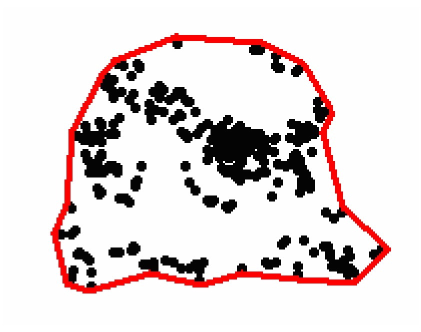

Figure 3.3.7 Alpha shape of Figure 3.3.2 (a)

3.3.2 Object Mask Generation Utilizing the Convex Hull Algorithm

Since the object mask generation is applied throughout the image sequence, the most critical requirement is the stability of the algorithm. In other words, the algorithm must generate object masks with minimum error when applied on all images, not only one image.

Recalling the octree reconstruction process mentioned in Chapter 2, the object model is reconstructed by eliminating the non-object part. When reconstructing model from object mask, the over-definition part in one mask may be eliminated in other masks, while the over-eliminated part is removed from the model and can not be recovered again. Therefore, the major requirement for the mask generation algorithm is to generate masks rather over-defined than over-eliminated.

Reviewing the three algorithms described in Section 3.3.1 according to the requirement. The alpha shape algorithm is inapplicable for the reason that, there is no method in defining proper circle radius *r* without human's aid. The try and error method is not suitable when processing images one by one. If applying a fixed predefined circle radius *r* instead of optimizing it for each image, the generated object mask might be over-eliminated if holes are generated due to some misjudgments. On the other hand, the concave hull algorithm is also not adaptable for the fact that the algorithm might misjudge the concave and result in incomplete object masks. These two algorithms both have the possibility of over-eliminating the object mask. Thus, the two algorithms are both not suitable for the situation concerned. Therefore, the best and only choice for the object mask generation algorithm is the convex hull algorithm.

The object mask generation step of the proposed background removal algorithm implements the gift wrapping algorithm [9], one of the convex hull algorithms. The main concept of gift wrapping algorithm is to wrap all points with a foldable line. Starting from the leftmost point p_0, a line segment l_{01} is linked to the point p_1 which makes all other points lie right to the line segment l_{01}. Then start from p_1 and link a line segment l_{12} to next point p_2 which makes all other points lie right to l_{12}, and so forth. The algorithm terminates when line segment is liked back to starting point p_0, thus all points are wrapped inside a polygon composed of line segments, l_{01}, l_{12}, …, l_{m0} for some m.

The concept and executing process of gift wrapping algorithm are simple except the step for determining if all points lie right to a line segment. Instead of searching all possible line segments, the determination process can be simplified just by utilizing the angle relationship of line segment sequence. By setting a vertical pseudo line passing the leftmost point for initial judgment, the line segment with minimum clockwise included angle to the previous line segment will make all points lie to its right. Hence, the gift wrapping algorithm is transformed into linking line segments in sequence, as shown in Figure 3.3.8(a). In fact, the judgment can be further simplified into an equivalent situation of finding the line segment having minimum included angle with the elongation of previous line segment, as shown in Figure 3.3.8(b).

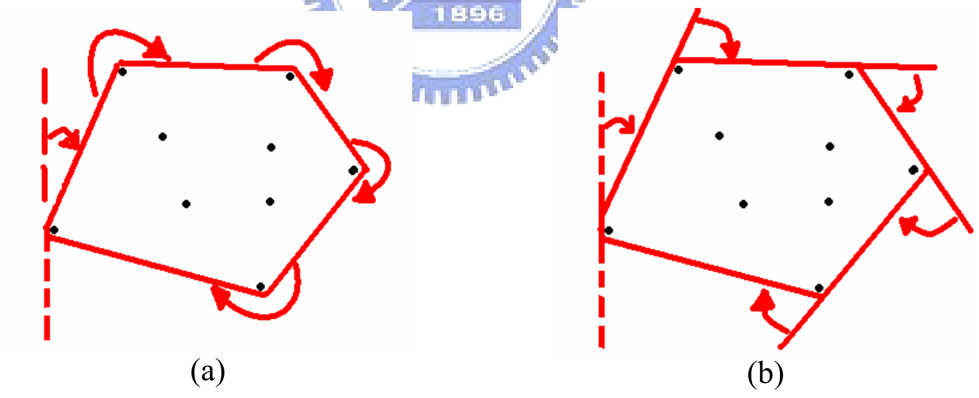

Figure 3.3.8 Process of gift wrapping algorithm. The dashed line indicates the pseudo line for initial judgment. [25]

To fulfill the previous fact that the next line segment is chosen based on the minimum clockwise included angle, it is easier and faster to check all the cosine values instead of calculating the actual angles, since the larger cosine value implies the smaller actual angle. For the example in Figure 3.3.9, the cosine value of the included angle θ between line segments l_{ij} and l_{jk} can be obtained as

$$cos(\theta)=\frac{\overrightarrow{p_i p_j}\cdot\overrightarrow{p_j p_k}}{\left|\overrightarrow{p_i p_j}\right|\left|\overrightarrow{p_j p_k}\right|} \tag{3.3.1}$$

where $\overrightarrow{p_i p_j}$ and $\overrightarrow{p_j p_k}$ are vectors formed by points (p_i, p_j) and points (p_j, p_k).

By applying (3.3.1) to all points, the point with the maximum $cos(\theta)$ can be found. Following the same procedure, the convex hull is formed, and the object mask can be generated by filling the inside of convex hull with gray-level value 255, as shown in Figure 3.3.10. The object mask generation algorithm is presented in Table 3.3.1.

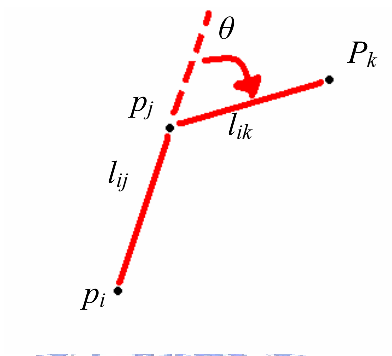

Figure 3.3.9 Reduced judgment situation

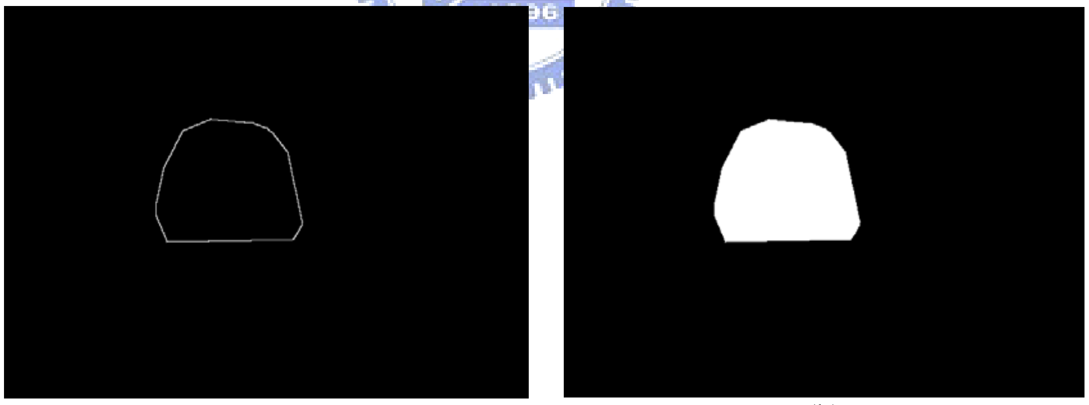

Figure 3.3.10 Object extraction operation : (a) convex hull of Figure 3.3. 1(b), (b) object mask of Figure 3.3. 1(b)

Table 3.3.1 Object mask generation algorithm

Input

A set of 2D points position ***P***.

Output

The object Mask *M* and an index array ***R*** indicates the order of points in ***P*** to form the convex hull

Algorithm

1. **Initialization** : Find the leftmost starting point and pseudo initial line

```
$\boldsymbol{R}[0] = 0$;
FOR EACH point $p_i$ in $\boldsymbol{P}$
    IF $p_i$ is to the left of $p_{\mathbf{R}[0]}$ THEN
        $\boldsymbol{R}[0] = i$;
    END IF
END FOR
$\vec{L} = (0, 1)$;
```

2. **Gift Wrapping** : Generate the convex hull

```
$j = 0$
DO
    $cos\theta = 0;$
    FOR EACH point $p_k$ in $\boldsymbol{P}$
        $cos\theta' = \dfrac{\vec{L} \bullet \overrightarrow{p_{\boldsymbol{R}[j]}p_k}}{\left|\vec{L}\right|\left|\overrightarrow{p_{\boldsymbol{R}[j]}p_k}\right|}$;
        IF $cos\theta < cos\theta'$ THEN
            $\boldsymbol{R}[j+1] = k$;
        END IF
    END FOR
    $\vec{L} = p_{\boldsymbol{R}[j+1]} - p_{\boldsymbol{R}[j]}$
    $j = j+1$;
WHILE $\boldsymbol{R}[j] <> \boldsymbol{R}[0]$
```

(Continued)

(Continued)

3. **Mask Generation** : Fill the inside of the convex hull

```
Generate a mask image $M$ at the same size of origin image
Fill $M$ with black
FOR $j$ = 1 to size($\boldsymbol{R}$)
    Draw white line on $M$ from $p_{\mathbf{R}[j-1]}$ to $p_{\mathbf{R}[j]}$
END FOR
Select a point $p_i$ where $i$ not in $\boldsymbol{R}$
Apply flood fill on $M$ start at $p_i$ to fill white
```

Chapter 4
Experiment Results

The experiment results of 3D model reconstruction by octree and object mask generation by convex hull are presented in Section 4.1 and 4.2, respectively. The reconstruction system prototype proposed in Figure 1.4 is implemented and the reconstruction result is shown in Section 4.3.

4.1 Octree Reconstruction Result

Experiments of octree reconstruction are based on the assumptions of ideal object mask image and object pose. Hence, the testing images of object silhouettes is generated by Maya PLE 8.5[2] instead of taking pictures in real world. The reason for generating testing image using 3D animation software such as Maya PLE 8.5 is that, object pose can be precisely controlled and the object silhouettes can be perfectly acquired through the software.

The 3D scene for generating test object silhouette image by Maya PLE 8.5 is designed and shown in Figure 4.1.1. A test object is specially designed with a hole to test if the octree can reconstruct an object model correctly for an object with holes. A camera is fixed in front of the object and facing it to capture test images, as shown in the bottom-left corner of Figure 4.1.1. Represented by five sets of brown arrows, five white lights are placed on the top, bottom, right, left, and in front of the test object. The object surface is set to fully reflect these lights. As a result, a white object in front of a black background makes an ideal object silhouette.

With the 3D scene, the object silhouette image is taken for every 10 degrees rotation of the object respect to the y axis. The resolution of the obtained silhouette images is 640x480 in pixel. Some silhouette images and corresponding object poses are shown in Figure 4.1.2.

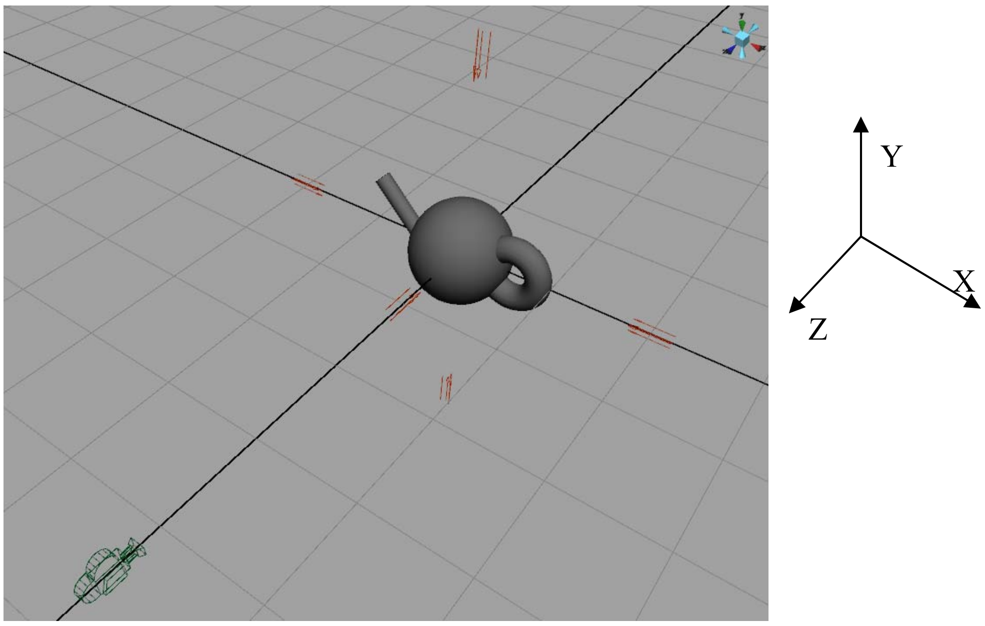

Figure 4.1.1 Arrangement for taking object silhouette image.

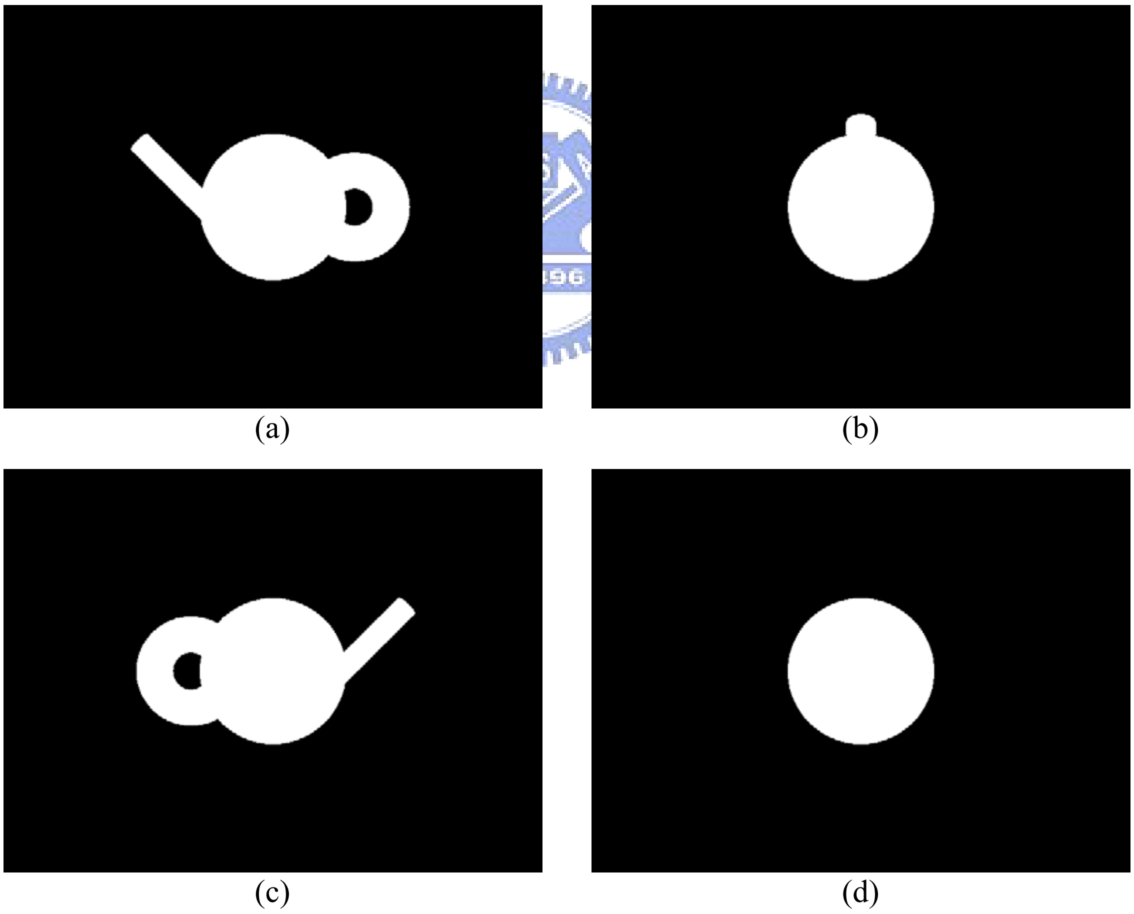

Figure 4.1.2 Object silhouettes generated by Maya PLE 8.5 : (a) object silhouette with rotation=(0°,0°,0°) and translation=(0,0,0), (b) object silhouette with rotation=(0°,90°,0°) and translation=(0,0,0), (c) object silhouette with rotation=(0°,180°,0°) and translation=(0,0,0), (d) object silhouette with rotation=(0°,270°,0°) and translation=(0,0,0)

As described in Chapter 2, the octree reconstruction process is based on shrinking of object model according to a set of continues images. Hence a test using only one image is performed first to make sure that the octree works correctly. The result of applying Figure 4.1.2(a) to the octree with max depth of 7 is shown in Figure 4.1.3. Figure 4.1.3 shows only the ON cubes determined after the octree process and the hole in Figure 4.1.2(a) is also reconstructed.

After making sure the octree process is functional, the experiment of applying all silhouette images to the octree is made. The whole process takes about 17 seconds with 36 630x480 silhouette images, and the reconstructed ON-cube model and triangulated model are shown in Figure 4.1.4 and Figure 4.1.5, respectively.

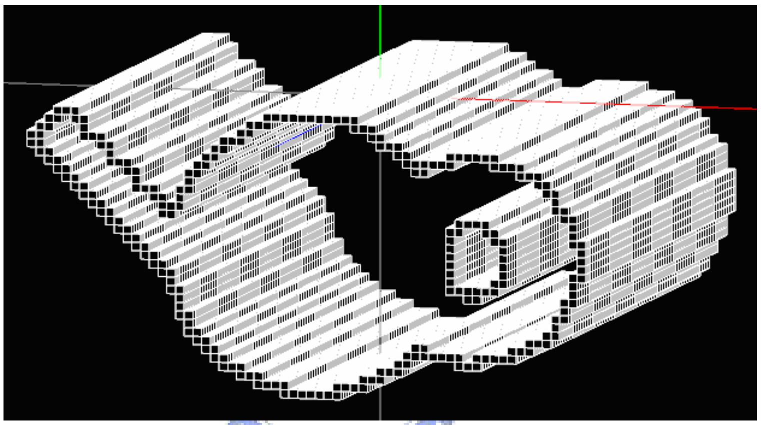
Figure 4.1.3 Reconstruction result of applying Figure 4.1.2(a) to the octree with max depth of 7. The figure shows only the ON cubes.

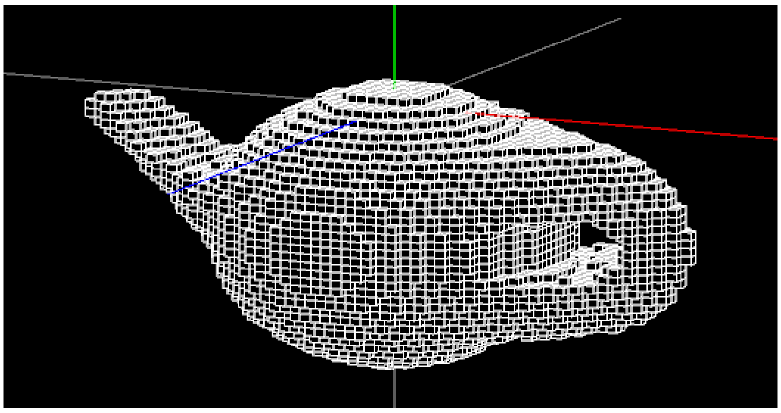
Figure 4.1.4 Reconstructed ON-cube model of input silhouette

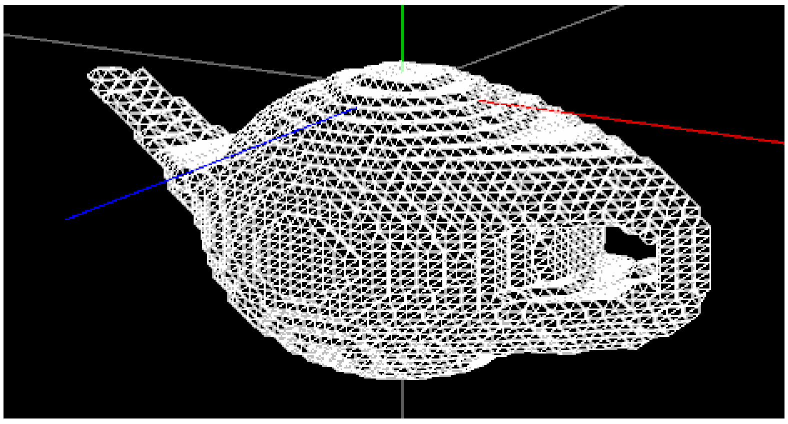

Figure 4.1.5 Triangulated model of applying triangulation process to the ON-cube model of Figure 4.1.4.

4.2 Background Removal Result

The experiment of background removal takes an image sequence of an object in unprepared environments as the input and generates the mask of the object. Mention that the proposed algorithm requires the 3D information of camera pose and feature point position, as described in Chapter 3. Hence, a camera tracker is required as a pre-process for the mask generation process. The voodoo camera tracker [21] is adopted to perform the task of reconstruction 3D camera poses and feature point positions of the image sequence. The result from voodoo camera tracker is then applied to the proposed algorithm to generate the mask of foreground object. The test image sequences are taken by a handheld free-motion camera, orbiting around and aiming at the target object. Meanwhile, the target object for each test is placed in an unprepared environment. The algorithm applies to two different cases for testing the performance.

4.2.1 The Toy-on-Table Sequence

The first experiment, referring to the scene of a small scale, is shooting at a toy placed on a table in a range about 60 degrees, called the Toy-on-Table sequence. The sequence contains 134 frames with frame rate of 30 fps, resolution of 640x480 in pixel, and color depth of 24-bit. Three frame of the sequence are shown in the "Original frame" column of Figure 4.2.1, while the corresponding foreground feature point distribution and background removal results are shown in the "Separated foreground feature points" column and "Background removal result" column of Figure 4.2.1, respectively. For the more detailed results, result of the foreground/background separation step, the first step of proposed algorithm, is shown in Figure 4.2.2. The whole process takes about 82 minutes to run on a laptop with Intel T2500 2.0GHz CPU and 512MB ram of DDRII-667. However, 81 minutes, or 98.7% of the execution time is spend on the camera tracking process.

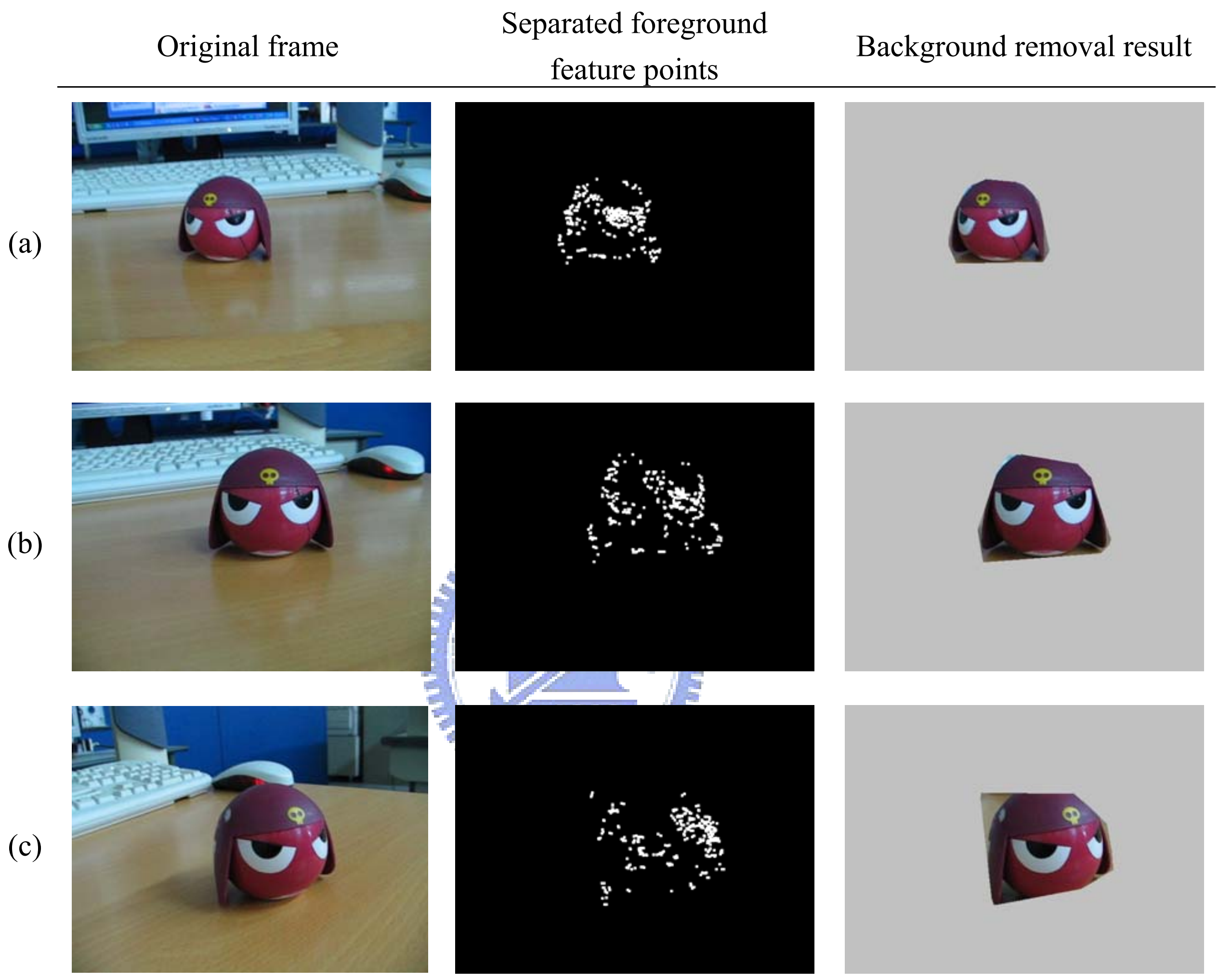
Original frame
Separated foreground feature points
Background removal result
(a)
(b)
(c)

Figure 4.2.1 Frames of the Toy-on-Table sequence, the background of "Background removal result" is changed to gray for clear view: (a) frame #0, (b) frame #67, (c) frame #134

Observing Figure 4.2.1, the performance of background removal varies from (a) to (c). The background removal extracts the object precisely in Figure 4.2.1(a), but shows non-ignorable large error in Figure 4.2.1(c). To probe into the causes of the error, three factors are found besides the one of adopting convex hull algorithm mentioned in Chapter 3. The first factor is the error of foreground/background separation. Because the object mask is directly generated by the foreground points, any error in foreground point determination may directly interferes the object mask result, which is clearly seen in the "Separated foreground feature point" image of Figure 4.2.1(c). The second factor is the lost of feature points on edges due to the detection algorithm adopted by the voodoo camera tracker. The above two factors mainly influence on the background removal result.

The last factor is caused by the error of the camera tracker's estimation process, including errors in 3D feature point positions, 3D camera positions, and 3D camera orientations. Note that this factor does not generate clear influence on the result when compared to the above two factors, for the reasons that the estimation error is small and the projection from 3D onto 2D reduces the error. Viewing Figure 4.2.2 again, the error in 3D feature point positions can be judged from two parts. The first part can be seen as some yellow points around the foreground object points marked in green, circled by orange ellipse, which should not exist in the original image. The second error is the yellow points in the left part, circled by red ellipse. Notice that the input image sequence was taken around the object in about 60 degrees, there should be no such points on the left part since no information were provided.

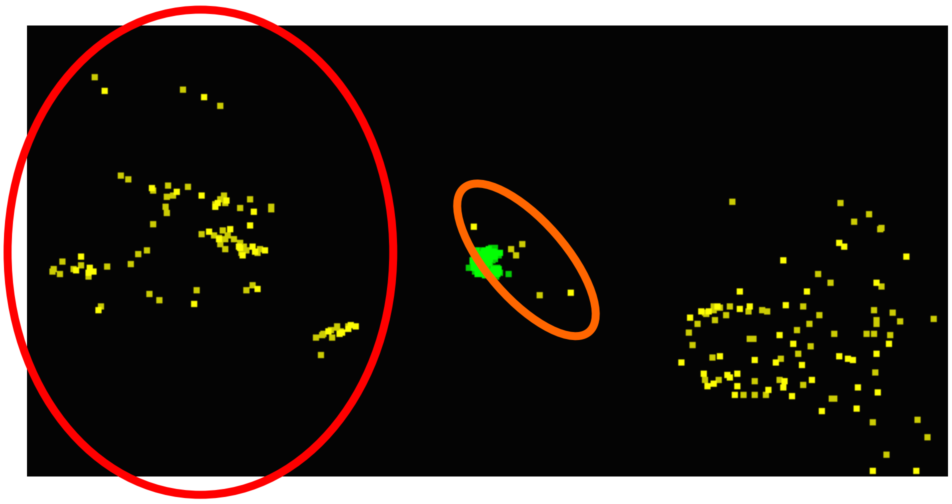

Figure 4.2.2 Foreground/background separation result the Toy-on-Table sequence. The green points are the foreground points and the yellow points are the background points.

4.2.2 The Statue Sequence

Different from the indoor condition in 4.2.1, the scale of scene of outdoor condition is larger than the indoor condition. The Statue sequence is the test sequence taking outdoor shooting at a statue in an open environment in a range about 70 degrees. The Statue sequence is an image sequence of 60 frames with frame rate of 15 fps, resolution of 640x480 in pixel, and color depth of 24-bit. Same as Section 4.2.1, Figure 4.2.3 shows the result of three frame of the Statue sequence and Figure 4.2.4 shows the result of the foreground/background separation step. Also mentioned that the whole process takes about 9 minutes to run on a laptop with Intel T2500 2.0GHz CPU and 512MB ram of DDRII-667. Also the same as Section 4.2.1, 8 minutes, or 88.9%, of the processing time is spent on the camera tracking process.

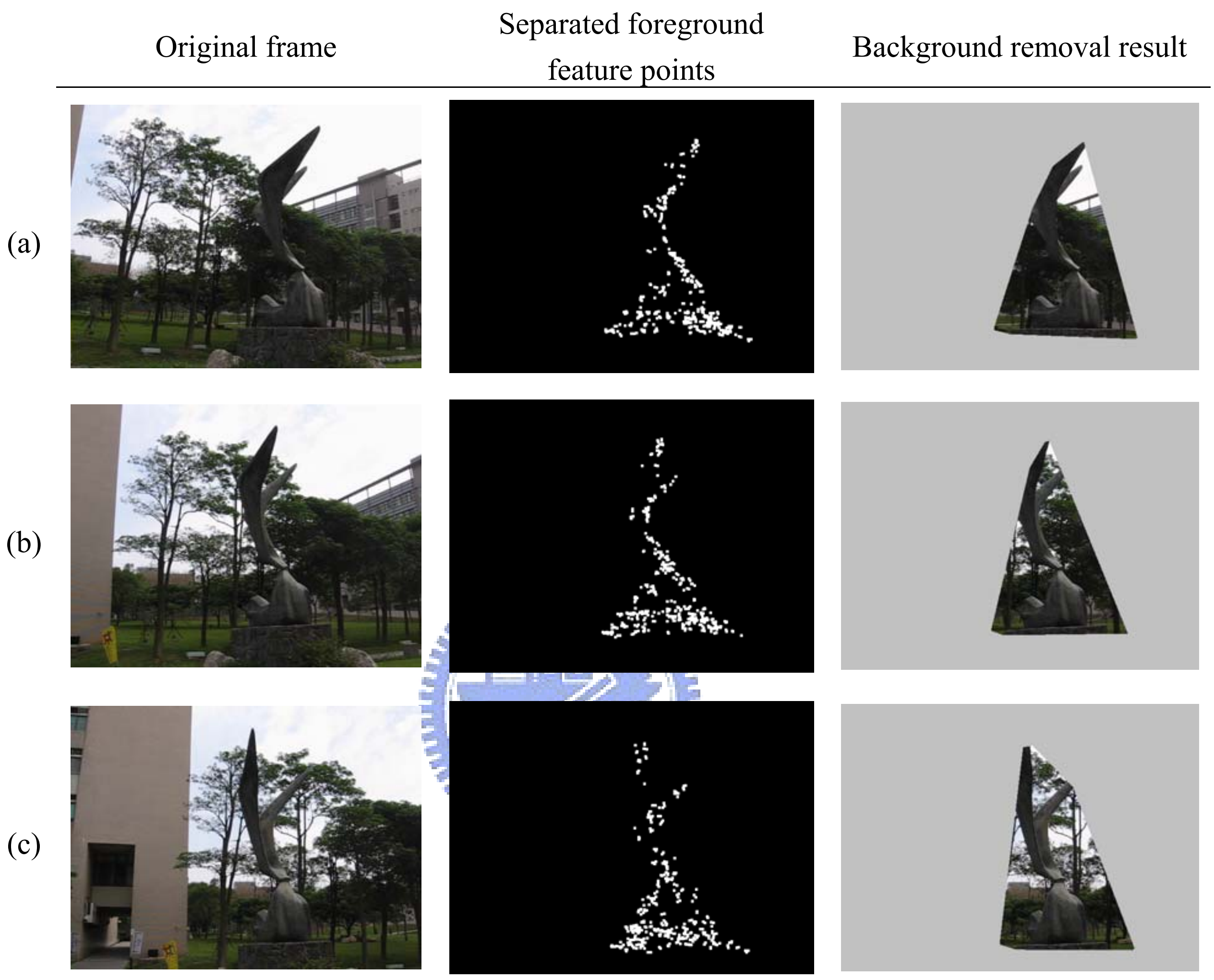
Original frame
Separated foreground feature points
Background removal result
(a)
(b)
(c)

Figure 4.2.3 Frames of the Statue sequence, the background of "Background removal result" is changed to gray for clear view: (a) frame #0, (b) frame #30, (c) frame #60

Comparing to Section 4.1, the most obvious difference is the background result affected by the shape of target object. For instance, since the object shape in Figure 4.2.1 is approximately a convex, the convex hull algorithm successfully generates object masks close to the object shape. However, when the convex hull algorithm applies to an object not solely composed of convex shape, as shown in Figure 4.2.3, it fails to recover the concave parts of the object. Even though the foreground separation result reveals the shape of the target object, the background removal result contains a large part of background due to the convex hull algorithm. Furthermore, Figure 4.2.4 also shows the serious influence of missing feature points on edges even though more feature points are detected than Figure 4.2.2.

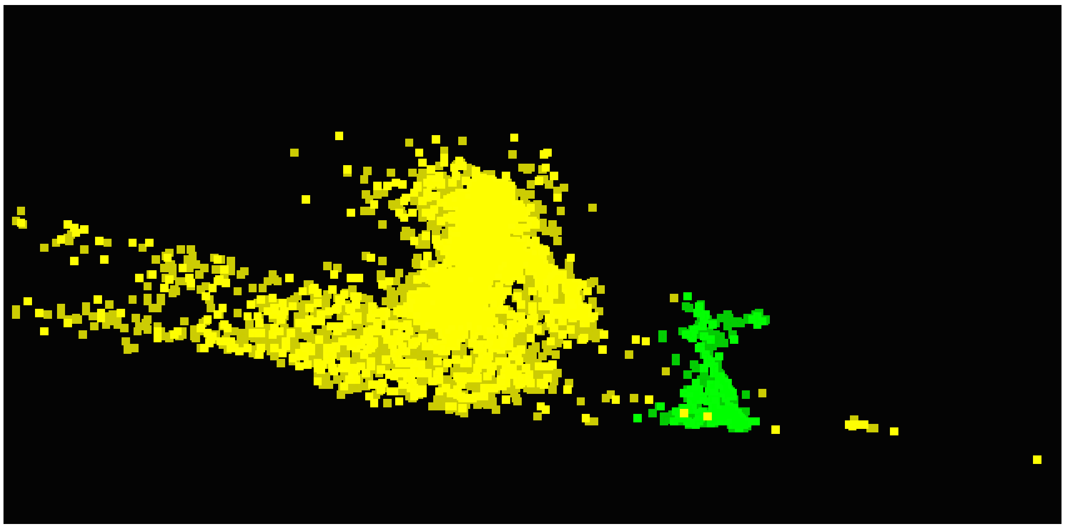

Figure 4.2.4 Foreground/background separation result of the Statue sequence. The green points are the foreground points and the yellow points are the background points.

4.3 3D Model Reconstruction System Result

Combining camera tracker, octree and proposed background removal algorithm, a new 3D model reconstruction system prototype is proposed, as mentioned in Section 1.3. However, after testing several available camera trackers including voodoo camera tracker 0.9.1 beta[21], PFTrack 4.0 evalution[34], ICARUS v2.09 personal edition[17], and SynthEyes Demo[1], a limitation of these camera trackers has been revealed. These camera trackers can only reconstruct information from image sequences taken by camera with change in viewing angle less than about 80 degrees. The limitation comes from the assumption of camera trackers that most of the feature points detected must remain in sight. Camera trackers reconstruct 3D information based on tracking the position change of these feature points. Once the change in viewing angle is larger than about 80 degrees, feature points in some frame may be quite different from another frame. Yet the camera tracker still tries to track feature points that are already lost and find a best solution of the 3D camera pose and feature point positions. As a result, the reconstructed information is totally collapsed due to the invalid tracking.

Observing Figure 1.3.1, the reconstructed 3D information from camera tracker is crucial for the proposed reconstruction system. The proposed reconstruction system can not function effectively without information from 360-degree object image sequences due to the camera tracker limitation. Instead, scale-down experiments are adopted to verify the reconstruction system. With the use of the Toy-on-table and Statue sequences in Section 4.2, the reconstruction results are shown in Figure 4.3.1 and 4.3.2 respectively.

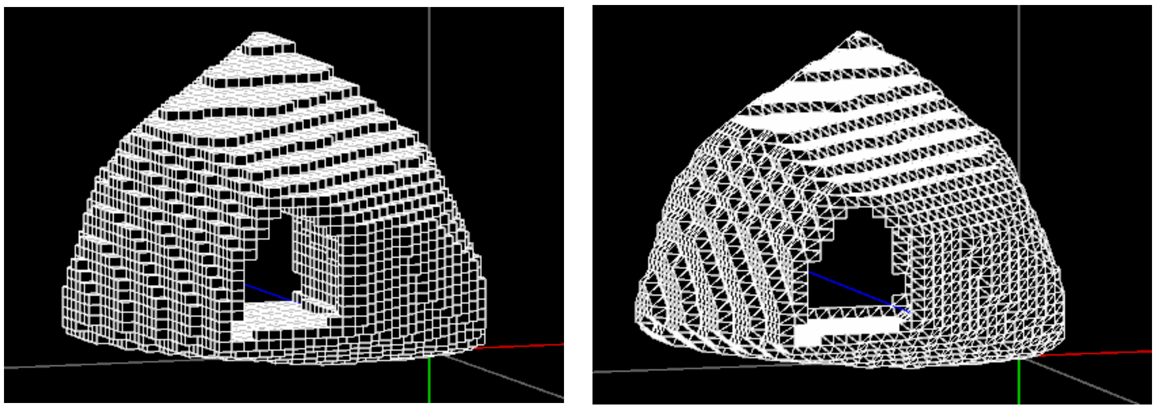
(a) (b)

Figure 4.3.1 Reconstruction result of Toy-on-table sequence. (a) octree model, (b) triangular model.

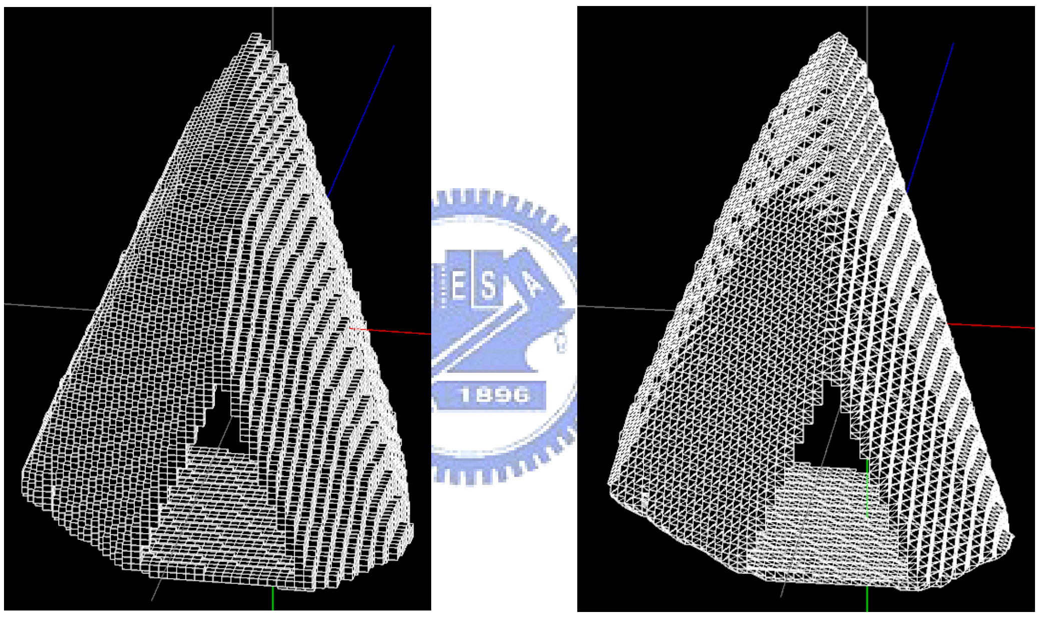
(a) (b)

Figure 4.3.2 Reconstruction result of Statue sequence. (a) octree model, (b) triangular model.

Viewing from Figure 4.3.1 and Figure 4.3.2, shapes of reconstructed models are much different from the target objects. The models are hollow since the results show only ON cubes and the hollow parts inside the surface actually represent undetermined cubes identified as IN cubes. The dissimilarity between object models and target objects is caused by two factors. The first factor is the object mask generated by the convex hull algorithm, as mentioned in Section 4.2, which is adopted to preserve the largest possible shape of the object. However, the concave part of the object shape is ignored at the same time, so that the reconstructed models can not reveal the concave surfaces. The second factor is the lack of 3D information.

Since the image sequence provides only 80 degrees of view, the available information is limited in the provided viewing range. Hence, the octree algorithm can only trim the 3D model in the range and then leads to an incomplete model.

Chapter 5
Conclusion

5.1 Conclusion

This thesis proposes a new prototype of 3D object reconstruction system dealing with more general cases. However, a 3D object reconstruction system requires knowledge and technologies of many aspects that can not be all mastered during the research period. Hence, the implementation of proposed system integrates some existing algorithm and programs, including the octree algorithm, the convex hull algorithm, and the voodoo camera tracker mentioned in Chapter 2, Section 3.3, and Section 4.2, respectively. Since the performance and stability of these integrated parts are verified by many researches and applications, the reconstruction system only focuses on the performance of background removal algorithm mentioned in Chapter 3.

Section 4.1 shows that the octree algorithm functions as rebuilding the 3D model from silhouettes and corresponding camera poses. Next in Section 4.2 shows the proposed background removal algorithm is workable, yet a lot to be improved. Unfortunately, experimental results in Section 4.3 reveal that the performance of the proposed system is restricted to the background removal algorithm and the camera tracker, especially the latter.

As mentioned above, this thesis focuses on proposing a brand new system dealing with problems never dealt before. This thesis proves the proposed algorithm and system structure are useful and many work can be done in the future to make the system better, as explained in the next section.

5.2 Future Work

5.2.1 The Camera Tracker

The limitation and effect of the existing camera tracker is presented in Section 4.3. Actually, the proposed algorithm can not fully work if the camera tracker limitation is not removed. This makes the camera tracker becomes the most major part in the proposed system to be improved. The experiment on different camera trackers shows that the limitation is a essential problem for all camera trackers. The problem should be a defect of the existing camera tracking algorithms. Hence, modifying the existing algorithm or designing a new algorithm for the shooting condition mentioned in this thesis is required.

The typical structure of a camera tracker is shown in Figure 5.2.1. The problem mentioned is caused by the feature point tracking block in the camera tracker. To deal with the problem, the feature point tracking algorithm must be improved to be able to determine whether a tracked point is appearing or disappearing in some frame. With the improvement, a point is tracked only when it is visible to the camera, not through out the image sequence. To achieve the requirement, the feature point tracking algorithm must track feature points based on not only the low level information such as edge, color, and texture, but also the information of higher level like geometry relationship. Hence, algorithms of higher information processing like image interpretation and understanding might be integrated into the point tracking algorithm.

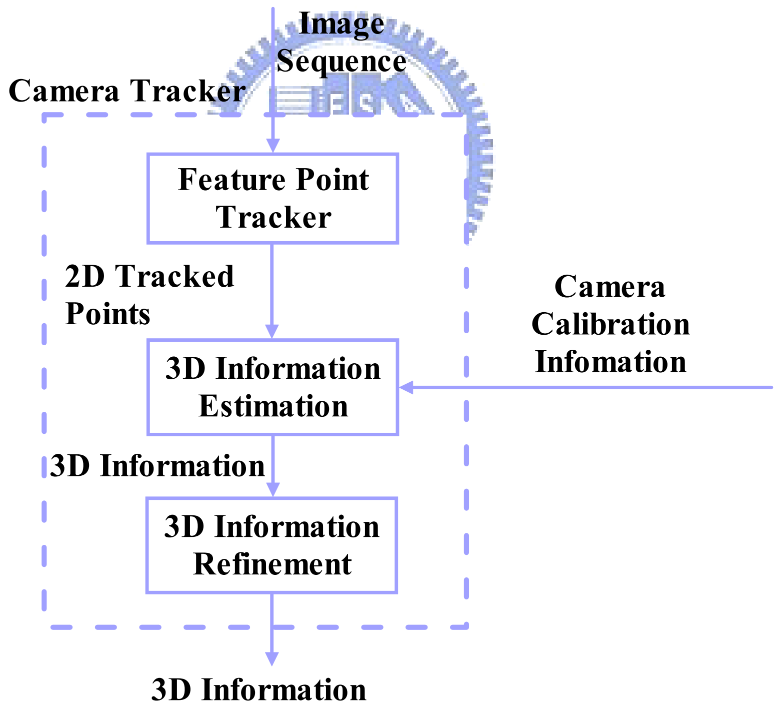

Figure 5.2.1 System structure for depth-map recovery algorithm

Besides the limitation of camera shooting angle, another minor improvement for the feature point tracker is also required for better background removal performance. As mentioned in Section 4.2, some important features including edges and corners of the object are not tracked by the feature point tracker. The lack of certain points makes the following process, the object mask generation in Section 3.3, becoming more complex and difficult to recover the shape of the object. In other words, more feature points provided, more accuracy the object mask becomes. Hence, the algorithm of feature point extraction could be improved to obtain more feature points on the object surface to support the background removal algorithm.

5.2.2 The Background Removal Algorithm

The background removal algorithm is composed of two steps. Currently, each step implements simple algorithm and works as a prototype. For the foreground/background separation step, the current algorithm applies a modified nearest neighbor algorithm on 3D point positions to separate points. The algorithm is fast and simple but not accurate enough. The separation step should imports information, such as texture, edge, and shape segment, from the object image to assist the separation algorithm. These information could be useful to eliminate points which are near but do not belong to the object.

On the other hand, the object mask generation step must be improved to be capable of dealing with concave contour and holes. From Section 4.2, it is clear that the ability is crucial for the quality of reconstructed model. However, it is difficult to determine the concave contour and holes of the object even with sufficient 2D point information. To obtain the accuracy object mask, information including texture and edges from object images must be taken into consideration to the object mask generation, same as the previous step.

5.2.3 The Texture Mapping Block

Though the texture mapping is a well-developed algorithm, the algorithm is much complex than octree algorithm and takes much time to implement. Hence, the texture mapping block is not implemented in the current system. However, a 3D model reconstruction system is incomplete without the texture mapping process, as a 3D model is incomplete without texture. The texture mapping block must be implemented in the future.

Bibliography

[1] Anderson Technologies LLC, "SynthEyes 2007 Demo," in http://www.ssontech.com/, 2007.

[2] Autodesk, "Maya Personal Learning Edition 8.5," in http://usa.autodesk.com/adsk/servlet/index?siteID=123112&id=7639525.

[3] P. A. Beardsley, P. H. S. Torr, and A. Zisserman, "3D Model Acquisition from Extended Image Sequences," in *ECCV '96 : Proc. 4th European Conference on Computer Vision-Volume II*, 1996, pp. 683-695.

[4] E. Boyer, "Object Models from Contour Sequences," in *ECCV '96 : Proc. 4th European Conference on Computer Vision-Volume II*, London, UK, 1996, pp. 109-118.

[5] M. S. Brown and W. B. Seales, "Beyond 2D images: effective 3D imaging for library materials," in *DL '00: Proceedings of the fifth ACM conference on Digital libraries*, New York, NY, USA, 2000, pp. 27-36.

[6] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3D objects with radial basis functions," in *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 2001, pp. 67-76.

[7] CGAL Open Source Project, "Chapter 24 : 2D Alpha Shapes," 2006.

[8] C. M. Christoudias, B. Georgescu, and P. Meer, "Synergism in Low Level Vision," *Porc. of 16th International Conference on Pattern Recognition,* vol. 04, pp. 150-155, 2002.

[9] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms, Second Edition*: {The MIT Press}, 2001.

[10] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1996, pp. 303-312.

[11] P. Eisert, E. Steinbach, and B. Girod, "Automatic reconstruction of stationary 3-D objects from multiple uncalibrated camera views," *Circuits and Systems for Video Technology, IEEE Transactions on,* vol. 10, pp. 261-277, 2000.

[12] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient Graph-Based Image Segmentation," *Int. J. Comput. Vision,* vol. 59, pp. 167-181, September 2004.

[13] K. Fischer, "Introduction to alpha shapes," 2000.

[14] A. W. Fitzgibbon, G. Cross, and A. Zisserman, "Automatic 3D Model Construction for Turn-Table Sequences," in *Proceedings of SMILE Workshop on Structure from Multiple Images in Large Scale Environments; Lecture Notes in Computer Science*, 1998, pp. 154-170.

[15] S. F. Frisken and R. N. Perry, "Efficient estimation of 3D Euclidean distance fields from 2D range images," in *VVS '02: Proceedings of the 2002 IEEE symposium on Volume visualization and graphics*, Piscataway, NJ, USA, 2002, pp. 81-88.

[16] B. Georgescu and C. M. Christoudias, "Edge Detection and Image SegmentatiON (EDISON) System," in http://www.caip.rutgers.edu/riul/research/robust.html, 2003.

[17] S. Gibson, J. Cook, T. Howard, R. Hubbold, and D. Oram, "ICARUS v2.09 personal edition," 2003.

[18] K. Higuchi, M. Hebert, and K. Ikeuchi, "Building 3-D models from unregistered range images," *Graph.Models Image Process.,* vol. 57, pp. 315-333, 1995.

[19] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," *Int'l Conf. on Computer Vision,* pp. 321-331, 1987.

[20] R. Koch, M. Pollefeys, and L. J. V. Gool, "Multi Viewpoint Stereo from Uncalibrated Video Sequences," in *ECCV '98 : Proc. 5th European Conference on Computer Vision-Volume I*, London, UK, 1998, pp. 55-71.

[21] Laboratorium für Informationstechnologie University of Hannover, "voodoo camera tracker 0.9.1 beta," in http://www.digilab.uni-hannover.de/docs/manual.html, 2007.

[22] M. Levoy, K. Pulli, B. Curless, S. Rusinkiewicz, D. Koller, L. Pereira, M. Ginzton, S. Anderson, J. Davis, J. Ginsberg, J. Shade, and D. Fulk, "The digital Michelangelo project: 3D scanning of large statues," in *SIGGRAPH '00*, New York, NY, USA, 2000, pp. 131-144.

[23] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," in *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, New York, NY, USA, 1987, pp. 163-169.

[24] C. Montani, R. Scateni, and R. Scopigno, "A modified look-up table for implicit disambiguation of Marching Cubes," *The Visual Computer,* vol. 10, pp. 353-355, December 1994.

[25] A. Moreira and M. Santos, "Concave Hull: a k-nearest neighbours approach for the computation of the region occupied by a set of points," 2007, pp. 8-11.

[26] C. Pheatt, J. Ballester, and D. Wilhelmi, "Low-cost three-dimensional scanning using range imaging," *J.Comput.Small Coll.,* vol. 20, pp. 13-19, 2005.

[27] M. Pollefeys, R. Koch, M. Vergauwen, and L. V. Gool, "Flexible acquisition of 3D structure from motion," in *Proc. 10th IMDSP Workshop*, 1998, pp. 90-95.

[28] L. Qingquan, L. Bijun, L. Yuguang, and C. Jing, "3D Modeling and Visualization Based on Laserscanning," *Geographic Information Sciences,* vol. 6, pp. 159-164, 12 2000.

[29] J. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision and Materials Sciences*: Cambridge University Press, 1999.

[30] S. Sullivan and J. Ponce, "Automatic Model Construction, Pose Estimation, and Object Recognition from Photographs Using Triangular Splines," in *ICCV '98: Proceedings of the Sixth International Conference on Computer Vision*, Washington, DC, USA, 1998, p. 510.

[31] B. Sumengen and B. S. Manjunath, "Graph Partitioning Active Contours (GPAC) for Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 28, p. 509, 2006.

[32] B. Sumner, "Active Contour Models: Snakes."

[33] R. Szeliski, "Rapid octree construction from image sequences," *CVGIP: Image Underst.,* vol. 58, pp. 23-32, 1993.

[34] The Pixel Farm, "PFTrack 4.0 evalution," 2007.

[35] N. Wolfgang and W. Jochen, "Automatic Reconstruction of 3D Objects using a Mobile Monoscopic Camera," in *3DIM '97 : Proc. Int. Conf. on Recent Advances in 3-D Digital Imaging and Modeling*, Washington, DC, USA, 1997, p. 173.

[36] C. Xu and J. L. Prince, "Snakes, Shapes, and Gradient Vector Flow," *IEEE Transactions on Image Processing,* vol. 7, pp. 359-369, March 1998.

[37] Y. Yemez and F. Schmitt, "3D reconstruction of real objects with high resolution shape and texture," *Image and Vision Computing,* vol. 22, pp. 1137-1153, NOV 1 2004.

[38] C. T. Zahn, "Graph-theoretic methods for detecting and describing gestalt clusters," *IEEE Transactions on Computing,* vol. 20, pp. 68-86, 1971.