FEATURE LINE EXTRACTION SYSTEM FOR PAINTER ROBOTS
お絵かきロボットのための特徴線抽出機構

by

Kenta Kitamoto
北元健太

A Senior Thesis
卒業論文

Submitted to
the Department of Information Science
the Faculty of Science, the University of Tokyo
on February 15, 2006
in Partial Fulfillment of the Requirements
for the Degree of Bachelor of Science

Thesis Supervisor: Katsushi Ikeuchi　池内克史
Professor of Information Science

ABSTRACT

Our laboratory is constructing a painter robot. The first step in painting is to extract desirable feature lines such as silhouette and boundary lines, which represent the object to be drawn. For this purpose, this thesis proposes a system that extracts feature lines from a textured 3D polygonal model.

First, the system builds a textured 3D model of the target object from multiple cameras using the technique of multiple-view geometry. Here we employ a graph-cut algorithm to refine the separation between the foreground and the background region in the captured images. As a result, the system can generate a high-quality 3D model. Then, the system extracts and combines two types of edges: geometry edges obtained from polygonal data, and photometry edges obtained from a rendered image. Both of them are observed from a virtual viewpoint.

We apply the proposed system to a textured 3D polygonal model of a human and verify that the system is effective to obtain appropriate feature lines for drawings.

論文要旨

我々の研究室ではヒューマノイドロボットに絵を描かせるというプロジェクトを進めている。このロボットにおいて、描画対象から書くべき線を抽出することが１つの重要な要素となる。本研究では、お絵かきロボットがテクスチャ付き３次元ポリゴンデータを元に描画を行う際に必要となる、書くべき特徴線を抽出する機構を提案する。

本機構では３次元計測された対象物体に対し、ポリゴンデータの幾何情報から得られる幾何エッジと、レンダリング結果の２次元画像から検出される画像エッジを組み合わせ、任意の仮想視点から見た場合の特徴線を提示する。この２種類のエッジの抽出のために３次元モデルを生成する際、多視点からのカメラ画像に対しグラフカット計算を用いた前景・背景判定を施すことで、より精密な３次元モデルを生成する。３次元モデルの精度の向上により幾何エッジの抽出精度が向上し、またテクスチャマッピングの精度も向上するため画像エッジの抽出精度も向上することが期待される。

本論文では実際に人物のテクスチャ付き３次元ポリゴンデータに対して本機構を適用し、ロボットによる描画に適した特徴線が得られたことを報告する。

Acknowledgements

First of all, I would like to thank Prof. Katsushi Ikeuchi for giving me the opportunity to work on our system and for being my advisor. I also thank to Dr. Koichi Ogawara for giving me a great deal of knowledge and materials, and proofreading this thesis. In addition, I'm grateful to all members of the Ikeuchi laboratory for their encouragement.

Contents

List of Figures

Chapter 1

Introduction

Recently, robots have accomplished a wide variety of tasks in many fields. They are expected to reduce human labor drastically in the area of industry, and the main purpose in developing robots is for practical use. Also, they have begun to play an active part in the area of entertainment and are performing in many places, such as expos and public institutions. Additionally, some researchers have focused on the development of artist robots, motivated by a wish to have lifeless things imitate human behavior.

In line with this trend, our laboratory is working on a painter robot project. The goal of this project is to construct a painter robot that can draw an oil painting of various objects: a human, a scene, a fruit arrangement, and so on. We aim at realizing not only drawing but also painting. The origin of our motivation is the analysis-by-synthesis of painting processes of human, and we aim not only at drawing by a robot but also at extracting general processes of painting action.

In this thesis we propose a system that extracts feature lines from data of object to be drawn. When we let the robot describe a line drawing, we have to give it feature lines to be drawn, such as silhouette and boundary lines, which represent the object. If we want oil paintings or watercolor paintings, the robot needs designation of area and color to be painted, in addition to feature lines. As the first step of this project, we start with line drawing.

1.1 Related Work and Our Robot

In the field of painter robots, drawing of human portraits has been the main thrust of study. Nakajima et al. built a portrait-drawing robot system that consists of image processing hardware that specializes in feature line detection from an image of a human face, and robot hardware to draw detected lines with a specially-made writing brush [1]. In recent attempts, Calinon et al. created a humanoid robot capable of drawing portraits with a quill pen through a human-like process of drawing [2]. While [1] attempted to draw portraits only by silhouette or boundary lines, [2] enabled their robot to fill the areas bounded by the contours. Both of these two creations use a humanoid robot and are for entertainment.

Among other types of painter robots, the application of robot teaching techniques to art has resulted in a calligrapher robot that learns brushwork from expert human calligraphers[3]. This robot consists of single robot arm, and is designed for the industrial use of putting pictures on ceramics. The PumaPaint Project[4] aims for networked robotics and employs painting as an experimental task; users can control the single-arm PUMA robot through the network by manipulating the location of brush strokes and the pressure the brush exerts on the canvas.

To summarize all the works described above, all of them prepare instructions to draw a picture for their robots by using some method, and the robots draw a picture by following the instructions. They cannot change their behavior during their drawing process according to the result of drawing. Moreover, most of them generate instructions from a 2D image, such as portrait photographs.

Our painter robot is shown in Figure 1.1 and 1.2. She is originally constructed to experiment a framework of ”learning from observation” through a variety of familiar tasks. In addition to drawing a picture, she is challenged to knot a rope[5], execute assembly tasks[6], and so on. She is designed to imitate the upper part of the human body, especially the eyes, arms, and multi-fingered hands. Given information of the picture to draw, she does it using visual feedback, obtained from her 9-eye stereo vision system, and force feedback, obtained from sensors attached to her hand; she draws a line along the given route until it reach an adequate length, seeing the picture she draws with her 9 eyes. So our painter robot is more robust than systems that draw a line by simply tracing a route generated in advance, and her drawing process is also more similar to the human process of drawing.

1.2 Drawing of 3D Object

Our interest on this project is not only in the drawing of a 2D image but also in drawing a 3D model. 2D image processing is much easier than dealing with a 3D object, and previous works on painter robots have tended to focus on representation based on 2D image analysis, as seen in [1, 2]. On the other hand, many recent studies on computer graphics have focused on non-photorealistic rendering of 3D objects [7, 8], in addition to those of 2D images [9, 10]. In this thesis, we take up a method to deal with a 3D object, which is more interesting than the processing of a 2D image. We are interested in processing a 3D object for two main reasons: high quality feature line extraction, and variation of viewpoints.

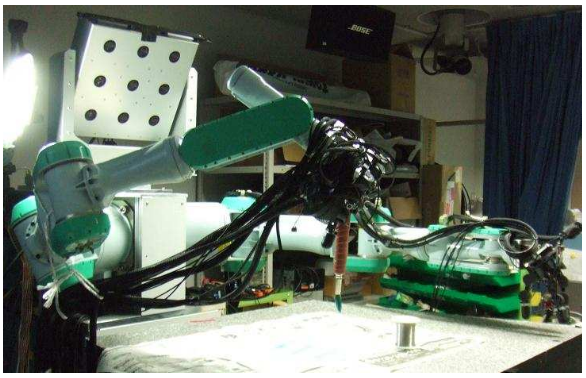


Figure 1.1: Our Painter Robot

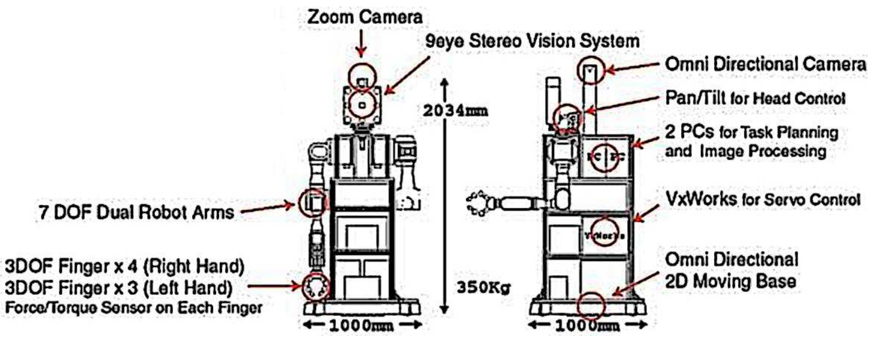

Zoom Camera
9eye Stereo Vision System
Omni Directional Camera
Pan/Tilt for Head Control
2 PCs for Task Planning and Image Processing
VxWorks for Servo Control
Omni Directional 2D Moving Base
7 DOF Dual Robot Arms
3DOF Finger x 4 (Right Hand)
3DOF Finger x 3 (Left Hand)
Force/Torque Sensor on Each Finger
2034mm
350Kg
1000mm
1000mm

Figure 1.2: Detail of Robot System

Feature line extraction from a 2D image is based on edge detection, and most approaches to edge detection utilize derivative techniques. The detected edges are interpreted as lines to be drawn, and compared with the model of the object, if needed. Of course, these methods are useful in drawing a 3D object, and we also utilize photometry edges as an important resource. However, there are some cases in which edge detection fails: an unclearly colored image, a monochrome object of complicated shape, overlapping objects of similar colors, and so on. We have difficulty in detecting edges from these objects because colors of pixels surrounding desirable edges resemble each other. This difficulty originates in poor quality of the image or limited information of the object obtained from a 2D image.

Geometric information obtained from 3D data solves the difficulty. By using geometric features, such as an object boundary and curvatures, we can obtain edges that couldn't be seen in 2D images. These geometric edges add more quality to edges detected from 2D images.

Also, 3D data enable us to see the target object from an arbitrary virtual viewpoint. We can see the object from unusual viewpoints, such as overhead heights, and a unique image of the object can be obtained. Furthermore, poor camerawork is supplemented with the aid of 3D technique. When we express the object by only one or more photographs, the object might be recognized unclearly because of an inadequate viewpoint, and this failure can't be recovered once we finished photographing. We can adjust the viewpoint by using a 3D technique, and recover poor camerawork in the process of making 3D data. Similarly, this flexibility of viewpoint also helps to obtain high-quality edges. It depends on a viewpoint whether an edge can be seen, and we can adjust the virtual viewpoint to the angle from which the desirable edges are seen clearly.

For all these reasons, dealing with a 3D object is effective in drawing by robots, and a high-quality 3D model is needed in this work. In this paper, we propose a framework to obtain a high-quality 3D model by using a graph-cut algorithm [11].

1.3 Thesis Contribution and Overview

This thesis describes our system, which realizes feature line extraction from reconstructed 3D model. By using this system, we can obtain a textured 3D polygonal model and feature lines to be drawn in an automated way. A 3D model is reconstructed only by providing this system with captured images and camera calibrations, and feature lines are calculated only by deciding a virtual viewpoint. All we have to do for painting by robots is to convert results into a format that a robot can accept. This automated system can be helpful in realizing drawing by a robot.

One characteristic approach performed in this thesis is refinement of silhouette extraction that is utilized in a series of 3D shape reconstruction. We employ graph-cut algorithm to give robustness to background subtraction, which is often used in computer vision and weak at slight environmental changes. This approach can be applied not only to 3D modeling but also to variety of tasks in computer vision, such as object recognition.

The process of this system is as follows:

1. Build a 3D model of the target object from multiple cameras
2. Map adequate textures on the model
3. Render the model
4. Extract and combine geometry and photometry edges

We introduce each step of our system in the following sections. First, in Chapter 2, we propose a method to build a 3D model by using the technique of multiple-view geometry, in which we employ a graph-cut algorithm to refine the model. Here we present the method to reconstruct surfaces of the model and map adequate textures on the obtained 3D model. Next in Chapter 3, we describe how to render the 3D model based on a virtual viewpoint selected arbitrarily and extract two types of edges: geometry and photometry edges.

After introducing our system, we show the experimental result of the system in Chapter 4. And finally in Chapter 5, we offer some conclusion and proposals for future work.

Chapter 2

3D Modeling

It is an old and difficult problem in computer vision to generate a 3D model of a scene given multiple 2D photographs taken of the scene. This problem applies to robot navigation, special effects for motion pictures, games, virtual reality, and so on. Because of its wide application, some broad classes of 3D reconstruction techniques have been developed.

One of the traditional methods for dealing with this problem, called *stereo vision* [12, 13, 14], is based on image matching. It calculates correspondences across 2D images on the basis of color intensity and/or image features, and then reconstructs the 3D shape by using triangulation and surface fitting. However, this approach has some disadvantages such as that:

- Viewpoints must be close together to avoid triangulation error.
- Correspondence across images may be lost spanning changes in viewpoint.
- There are no explicit solutions in handling occlusion differences.

Volume intersection [15, 16, 17], which is the method we employed for this project, is an alternative approach of 3D shape reconstruction. This method estimates the shape of 3D object from silhouettes in each 2D image, and overcomes some of the disadvantages of stereo vision as seen in Section 2.1. Although it has some disadvantages, combining other methods, such as *voxel coloring* [18], can solve some of them.

In this chapter, first we introduce the way to build a 3D model by using a volume intersection method and explain how the graph-cut algorithm contributes to this method. Then we show the method to reconstruct surfaces of the model as a polygon representation from its voxel representation and map textures on obtained surface planes.

2.1 Volume Intersection

Volume intersection uses a finite set of cameras, each of which contributes a viewpoint and a 2D image captured from that viewpoint, and gives a shape of the object by intersecting the volumes due to silhouettes on the 2D images, as shown in Figure 2.1.

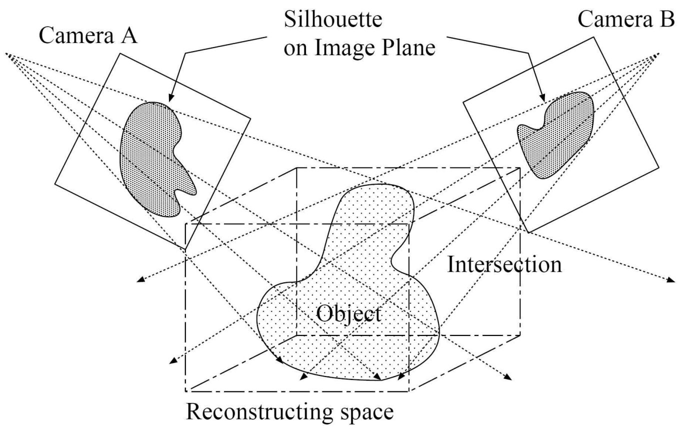


Figure 2.1: Silhouette Volume Intersection[19]

Suppose that a silhouette of the object S_i is given from each 2D image I_i. Then S_i gives a solid cone C_i by perspective projection of S_i from a corresponding viewpoint V_i. If parallel projection is used, C_i is a solid cylinder. Given that n images are available, the result of volume intersection R_n is obtained in this way:

$$R_n = \bigcap_{i=1}^{n} C_i \tag{2.1}$$

This technique is based on the fact that the object is included in a cone or cylinder formed by back-projecting a silhouette S_i of the object from a corresponding viewpoint V_i. The convex polyhedron R_n formed by this method is called *visual hull,* and means a maximal shape that gives the same silhouette as the actual object for all cameras [16]. The size of the visual hull decreases monotonically with the number of images n, and this decrease implies improvement of the quality. However, even when an infinite number of images is used, not all concavities can be modeled with a visual hull.

We chose the volume intersection method because of merits compared with other 3D reconstruction techniques:

- No restriction on unobserved parts

We can approximately reconstruct parts that are not observed in common by two or more cameras.

- No need to handle matching problem of points
  Matching problem is difficult if the target point is far away.

- Natural reconstruction of whole shape of the object
  Maximal shape is obtained even if uncertain parts remain.

- Possibility to take in voxel coloring
  Concave parts can be reconstructed by checking photo consistency of each voxel.

In this project, it is most important to reconstruct the whole shape of the object. Volume intersection method is superior for this purpose.

We start with a set of source images $\{I_i\}$ that are simply projections of the object onto n known image planes. Each of these n images must then be segmented into a binary image containing foreground regions to which the object projects; everything else is background. If these foreground regions are then back-projected into 3D space and intersected, the resultant volume is the inferred visual hull of the object. It is evident that higher quality of separation between the foreground and the background contributes to generate a better 3D model.

In implementing the volume intersection technique, we have to select one principle out of possible methods as below:

(a) Back-project contours of each silhouette as polygons and intersect polygons.

(b) Back-project all pixels of each silhouette as polygons and intersect polygons.

(c) Divide the space into voxels, and back-project all silhouettes checking whether each voxel is included by each of them.

(d) Divide the space into voxels, and project each voxel upon all image planes checking whether it is included by each silhouette.

The former two principles are polygon-based, and the latter two are voxel-based. Polygon-based methods are generally high quality because the quality is restricted only by the resolution of images. On the other hand, voxel-based methods are also restricted by the resolution of voxels. However, method (a) demands complicated processes, such as judging intersections of planes, and costs a lot of processing time. Processes of method (b) are somewhat simpler than (a), but needs excessive amounts of polygons and memory. Methods (c) and (d) are not so complicated because they are realized with loops of voxels and silhouettes, and necessary

```
foreach $v \in V$ begin
    $v :=$ true;
    foreach $i \in I$ begin
        $p :=$ pixel obtained by projecting $v$ onto the plane of $i$;
        if $p \notin$ silhouette in $i$ then
            $v :=$ false;
            break;
        endif
    end
end
```

Figure 2.2: Space Carving Method

V	:	set of all voxels
I	:	set of all images
`true`	:	v is in the visual hull
`false`	:	v is not in the visual hull

memory depends only on resolution of voxels. We employed method (d), which is shown in Figure 2.2 and called *space carving*[20], because of the ease of implementation.

Pursuing high resolution of voxels and efficiency of calculations, we combined the octree algorithm[17] with space carving. After finishing space carving for a set of voxels, each voxel included in the visual hull is divided into 8 voxels as seen in Figure 2.3. Then space carving is performed again, beginning from the new set of voxels, which consists only of divided voxels; voxels that are not included in the visual hull on the previous stage are eliminated.

Volume intersection includes the possibility to incorporate a voxel coloring method, although not implemented yet in our system. This method is based on photo consistency, i.e., that a voxel should have the same color as the corresponding pixels in each source image. The algorithm is as follows:

1. Take a voxel on the surface of a 3D model.
2. Back-project the voxel onto all images.
3. Check if the voxel satisfies photo consistency and delete it if not.
4. Repeat 1..3 until all surface voxels satisfy photo consistency

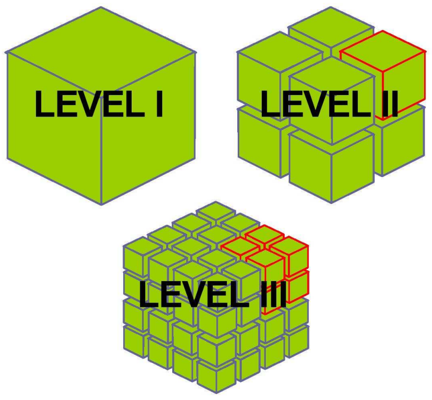


Figure 2.3: Octree Division

In step 3, photo consistency is judged by a statistical approach. For instance, by calculating variances ${\sigma_R}^2, {\sigma_G}^2, {\sigma_B}^2$ of the RGB colors obtained in step 2, summing them up to σ^2 (i.e., $\sigma^2 = {\sigma_R}^2 + {\sigma_G}^2 + {\sigma_B}^2$) and comparing with a given threshold τ (i.e., if $\sigma^2 < \tau$, the voxel is photo consistent), this judgment is performed.

2.2 Silhouette Extraction

Volume intersection needs silhouettes of the object, and quality of silhouettes affect The quality of the built 3D model. In this section, we propose a method to obtain high-quality silhouettes by using two methods: background subtraction and graph-cut image segmentation.

2.2.1 Background Subtraction

In order to achieve silhouette extraction, background subtraction technique is widely used. The basic scheme of background subtraction is to calculate the difference between two images: the current image that has the object in it and the reference image that models the background scene. The most simple type of this method utilizes two pictures taken from the same viewpoint: a foreground picture F that includes the target object and a background picture B that does not. Let p denote a pixel in both F and B, and $F(p), B(p)$ denote the properly encoded color value of the pixel p in F, B. If p satisfies the following inequality, then the foreground region contains p:

$$|F(p) - B(p)| > \tau \tag{2.2}$$

Here τ is a predefined threshold.

The above is a basic method, and many improved versions are proposed. The idea of this method is based on color consistency of the background region. This method is very simple and easy to automate. But the simplest method is too weak for fluctuations of the color of each pixel, such as a slight change of background and shadows due to the target object. To avoid these errors, we employed a statistical approach that utilizes a novel computational color model of *brightness distortion* and *chromaticity distortion* [21].

The employed method creates a statistical model for each pixel using a set of N background images. A pixel i in the reference image is modeled by a 4-tuple $< E_i, s_i, a_i, b_i >$. Each of them respectively denotes the expected color value, the standard deviation of color value, the variation of the brightness distortion, and the variation of the brightness distortion.

First, we calculate the most basic statistics of background images as follows:

$$E_i = [\mu_R(i), \mu_G(i), \mu_B(i)] \quad (2.3)$$

$$s_i = [\sigma_R(i), \sigma_G(i), \sigma_B(i)] \quad (2.4)$$

Here $mu_R(i), \mu_G(i), \mu_B(i)$ and $\sigma_R(i), \sigma_G(i), \sigma_B(i)$ are the arithmetic means and the standard deviations of the ith pixel's red, green, blue values.

Second, we calculate the brightness distortion α_i and color distortion CD_i corresponding to a current image that we want to subtract from the background. Let $I_i = [I_R(i), I_G(i), I_B(i)]$ denote the pixel i's RGB color value in the current image. Then,

$$\alpha_i = \frac{\dfrac{I_R(i)\mu_R(i)}{\sigma^2{}_R(i)} + \dfrac{I_G(i)\mu_G(i)}{\sigma^2{}_G(i)} + \dfrac{I_B(i)\mu_B(i)}{\sigma^2{}_B(i)}}{\left(\dfrac{\mu_R(i)}{\sigma_R(i)}\right)^2 + \left(\dfrac{\mu_G(i)}{\sigma_G(i)}\right)^2 + \left(\dfrac{\mu_B(i)}{\sigma_B(i)}\right)^2} \quad (2.5)$$

$$CD_i = \|I_i - \alpha_i E_I\| \quad (2.6)$$

Note that α_i minimizes $(I_i - \alpha_i E_i)^2$. As seen in Figure 2.4, the difference between I_i and E_i is decomposed into brightness (α_i) and chromaticity (CD_i). α_i means a ratio of brightness and CD_i means a distance of chromaticity.

Third, a_i and b_i introduced above is calculated as *root mean standard*(RMS):

$$a_i = \sqrt{\frac{\sum_{n=1}^{N} (\alpha_i - 1)^2}{N}} \quad (2.7)$$

$$b_i = \sqrt{\frac{\sum_{n=1}^{N} (CD_i)^2}{N}} \quad (2.8)$$

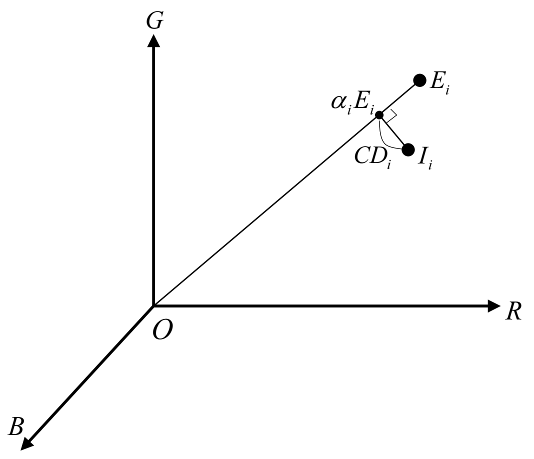


Figure 2.4: Color Model in the RGB color space

a_i and b_i are utilized to normalize α_i and CD_i. Since the different pixels yield different distributions of α_i and CD_i, they must be rescaled in order to use a single threshold for all of the pixels.

$$\widehat{\alpha_i} = \frac{\alpha_i - 1}{a_i} \tag{2.9}$$

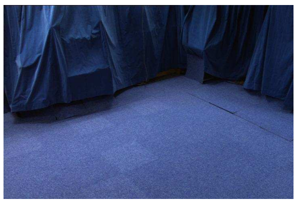
$$\widehat{CD_i} = \frac{CD_i}{b_i} \tag{2.10}$$

Finally, each pixel i is classified into four categories: B(Background), F(Foreground), S(Shaded background or shadow), and H(Highlighted background). The label $L(i)$ of a pixel i is determined by the following decision procedure:

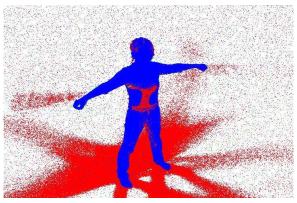
$$L(i) = \begin{cases} F \ : \ \widehat{CD_i} > \tau_{CD}, \quad else \\ B \ : \ \widehat{\alpha_i} < \tau_{\alpha 1} \ and \ \widehat{\alpha_i} > \tau_{\alpha 2}, \quad else \\ S \ : \ \widehat{\alpha_i} < 0, \quad else \\ H \ : \ otherwise \end{cases} \tag{2.11}$$

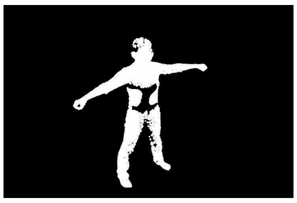
Here τ_{CD}, $\tau_{\alpha 1}$ and $\tau_{\alpha 2}$ are thresholds that can be selected by using a statistical learning procedure automatically based on a given *detection rate r*. r satisfies $0 < r < 1$ and means the rate of error that is ignored in the distribution.

In spite of these prudent procedures, results of this method is not always satisfactory for volume intersection because of a wrong boundary and/or holes in the silhouette, as seen in


(a) Background Picture

(b) Current Picture


(c) Labeling Result


(d) Silhouette

In (c), a pixel labeled as B,F,S,H is represented respectively by white, blue, red, green.

Figure 2.5: Result of Background Subtraction

Figure 2.5. We propose a novel method to solve this problem in the rest of this section.

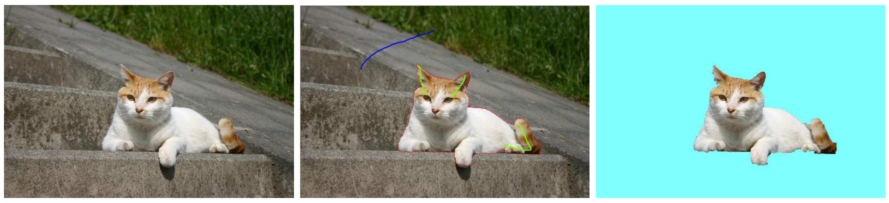
2.2.2 Graph-cut Image Segmentation

Image segmentation to foreground and background regions is formulated as a graph-cut problem in [11]. An image segmentation problem can be posed as a binary labeling problem. Suppose that the image is a graph $G = (V,E)$, where V is the set of all vertices and E is the set of all edges that connects adjacent vertices. In our system, V is the set of all small regions in the image obtained by pre-dividing pixels based on color similarity using the watershed algorithm[22], and E is the set of connections between adjacent regions. The labeling problem is to assign a unique label x_i for each vertex $i \in V$. Here $x_i \in \{0,1\}$; $x_i = 1$ means that i is foreground and $x_i = 0$ means background. The solution $X = \{x_i\}$ can be obtained by minimizing a Gibbs energy $E(X)$ [23]:

$$G(X) = \sum_{i \in V} G_1(x_i) + \sum_{(i,j) \in E} G_2(x_i, x_j) \tag{2.12}$$

where $G_1(x_i)$ is the likelihood energy, encoding the cost when the label of vertex i is x_i, and $G_2(x_i,x_j)$ is the prior energy, denoting the cost when the labels of adjacent vertices i and j are

x_i and x_j respectively. The specific form of $G_1(x_i)$ and $G_2(x_i, x_j)$ is given in [11], and some of $\{x_i\}$ must have been given as initial value to calculate $G_1(x_i)$. This energy minimization problem can be posed as a graph-cut problem and solved efficiently by using a technique described in [24].



(a) Original Image (b) Designation of foreground and background (c) Extracted Silhouette

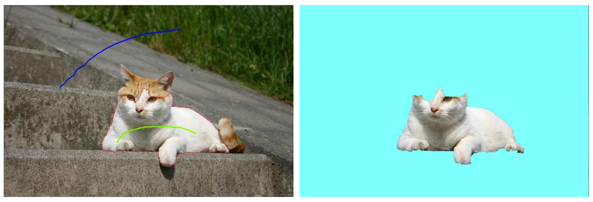
In (b), the foreground and background regions are designated with green lines and a blue line respectively, and the resultant boundary is shown by a red polygon.

Figure 2.6: Silhouette Extraction using Graph-cut

When we use this technique for image segmentation, we have to designate some pixels of foreground and background, as shown in Figure 2.6(b). Beginning with a lesser designation of pixels, this technique gives us a very fine separation of foreground and background region, as seen in Figure 2.6(c).

The merit of this method is that we can obtain a clear boundary between foreground and background regions. However, this method needs designation of some pixels of foreground and background initially, although it does not need much designation. This method fails if appropriate pixels of foreground and background are not given, as seen in Figure 2.7. In practical use, this designation of pixels must be given manually, intending a necessary foreground and unnecessary background region by seeing the image.

When we use the volume intersection method, we often use many images to improve the quality of the 3D model. It is too troublesome to apply the graph-cut algorithm manually to all used images. Besides, our system aims to automate the whole process of feature line extraction. For these reasons, we cannot support the use of the graph-cut algorithm in a manual way.


(a) Poor designation of foreground

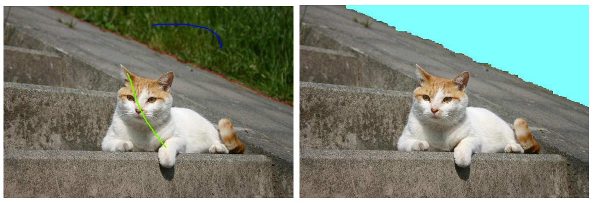

(b) Poor designation of background

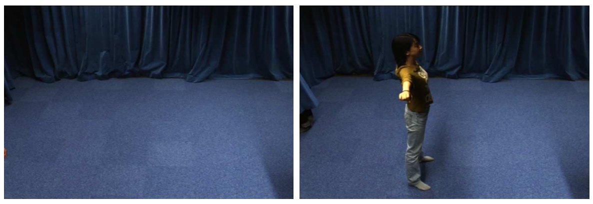
Figure 2.7: Failed Examples of Graph-cut Algorithm

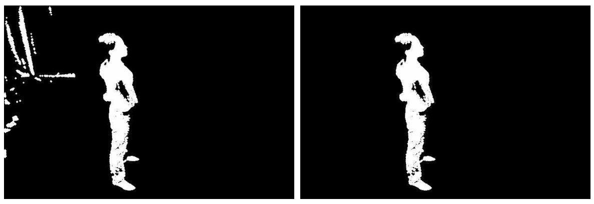
2.2.3 Combination of Two Techniques

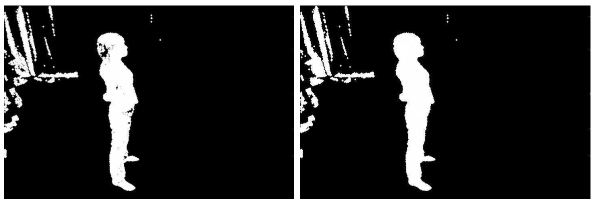
We propose combining the background subtraction method with the graph-cut algorithm. The proposed procedure is as follows:

1. Apply background subtraction twice to the target image using two different detection rates r_1, r_2. Here $r_1 > r_2$, and let S_1, S_2 denote resultant binary images corresponding to r_1, r_2.
2. Employ the foreground region in S_1 with a slight adjustment for designation as foreground in the graph-cut algorithm.
3. Employ the background region in S_2 with slight adjustment for designation as background in graph-cut algorithm.
4. Execute the graph-cut algorithm using the result above as an initial designation.

First, we apply a background subtraction method to the target image. Here the large detection rate r_1 results in a small and reliable foreground region. On the other hand, the small rate r_2 results in a large and uncertain foreground region. To reverse this, the background region is small and reliable. For example, r_1, r_2 can be set to 0.999 and 0.7 respectively.


(a) Original Images


(b) Foreground and Adjustment


(c) Background and Adjustment

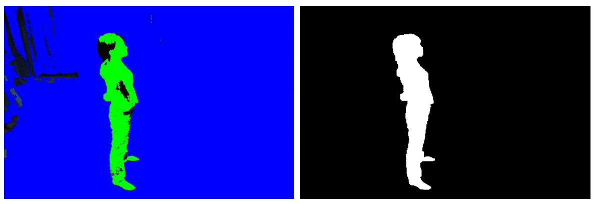

(d) Designation and Result

Figure 2.8: Result of Combined Silhouette Extraction

In the second step, the foreground region in the resulting image S_1 can include a slight error, such as some pixels outside the target object. Although the error pixels are few, they can largely influence a result of graph-cut algorithm. For that reason, we added a function to eliminate all small foreground regions that are not connected to the largest region. Also in the third step, we can eliminate all small background regions by a similar function. And finally, we can apply the graph-cut algorithm to the original image.

This combination of two techniques can make up for the defects of each technique. The background subtraction method automatically gives the initial designation to the graph-cut algorithm, and the graph-cut algorithm calculates finer separation than the background subtraction method. A fine silhouette of the object can be obtained by using this combined method and used in volume intersection.

2.3 Surface Reconstruction

The method described in the previous section gives us a 3D model of the target object as a set of voxels. Of course, we can render it as it is and texture on it. However, surfaces of a voxel-based 3D model are rugged and unnatural. Moreover, such surfaces may form wrong edges: a wrong normal vector of each plane causes a wrong geometry edge, and a rendered image with aliases causes the wrong photometry edges.

Marching cubes [25] is widely used in order to reconstruct a polygonal 3D model from a volumetric data set in which each voxel has scalar value. This method first binarizes the value of each voxel using a threshold; 0 is assigned if the voxel is outside the surface, and 1 is assigned if inside. In this system, our set of voxels is originally binary and no thresholds are needed. Then it considers cubes whose vertices are 8 adjacent voxels. Here note that the surface intersects those cubes where some vertices are 0 and the others are 1. The surface is reconstructed by checking the pattern of vertices in each cube.

The possible state of 8 vertices in a cube is $2^8 = 256$, and they can be reduced to 15 by using symmetry, such as rotation, mirroring, and state reversing. Therefore we can locate a surface in each cube by comparing the vertices' state with facets shown in Figure 2.9. Here a vertex of triangles is obtained by linear interpolation of a corresponding cube edge. In our system, the value of each voxel is binary and the interpolated point is just a midpoint.

Although marching cubes is a very practical and simple algorithm, it has a problem of topological inconsistency, i.e., there can be holes in created surfaces in some cases such as Figure 2.10. To overcome this defect, we employed facets introduced in [27]. Consequently, the surface of our volumetric model is represented as a polygon that consists of triangle patches.

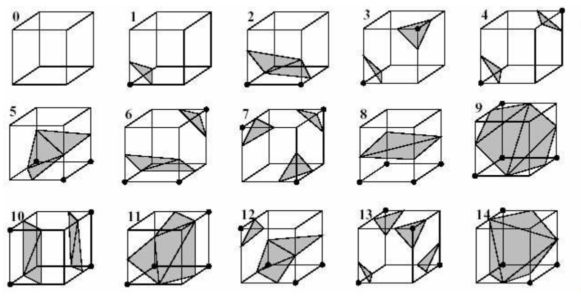


Figure 2.9: Set of Facets in Original Marching Cubes[26]

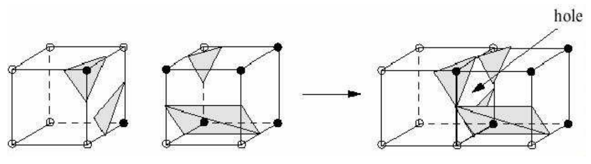


Figure 2.10: Topological Inconsistency[26]

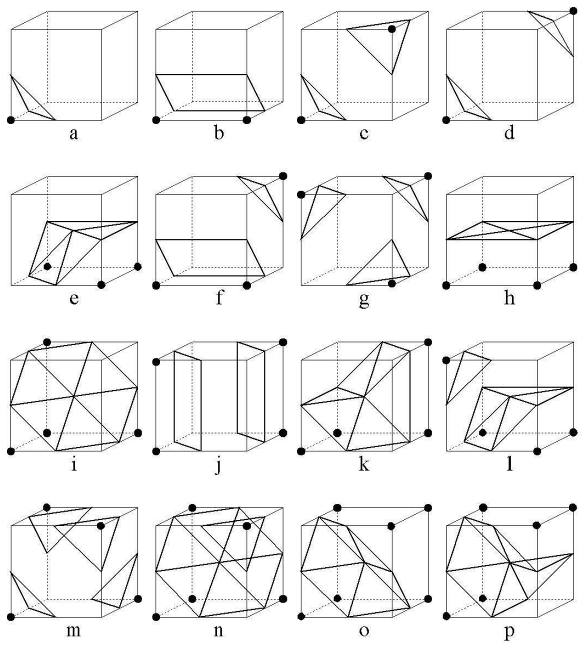


Figure 2.11: Employed Set of Facets[27]

2.4 Texture Mapping

Given a 3D polygonal model of the object, our 3D modeling is finalized by mapping a proper texture on each triangle patch. Here we have to determine which image is to be textured on a patch. It is natural that a patch is textured by the image that is taken by the camera facing the patch.

We solved this problem by using the normal of each patch and camera position. We apply this algorithm to each patch:

1. Calculate the normal vector $\mathbf{n}$ and normalized direction vectors $\mathbf{p_1}, ..., \mathbf{p_N}$ from the patch to all N camera positions $\mathbf{c_1}, ..., \mathbf{c_N}$.
2. Compute inner products $u_i = \mathbf{n} \cdot \mathbf{p_i} \quad (i = 1, ..., N)$.
3. Select I such that $u_I = \max\{u_i\}$ and texture this patch using the Ith image.

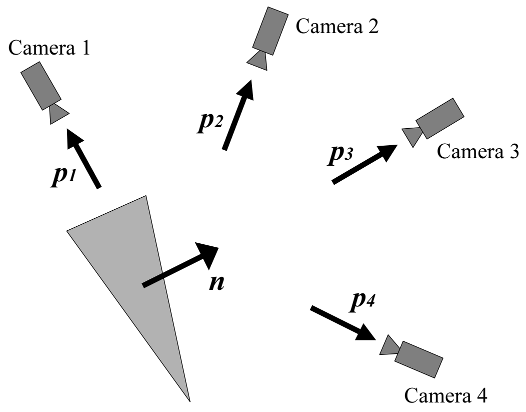


Figure 2.12: Camera Selection for Texturing

However, this algorithm fails in texturing patches hidden by another part. Z-buffer solves this visibility problem by removing unsuitable images. If a patch is hidden when seen from a camera, the image taken by the camera cannot be a candidate for texturing the patch.

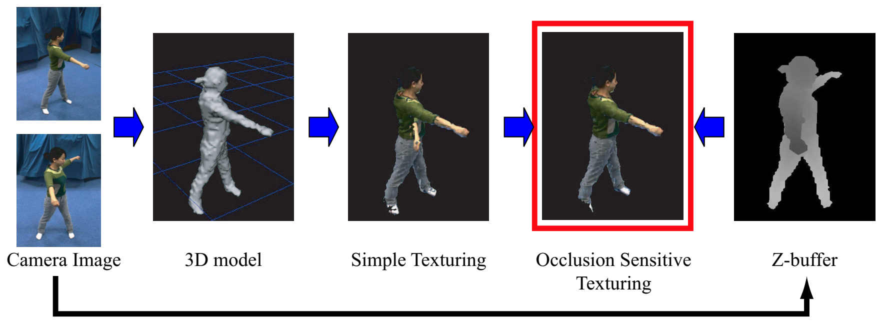


Figure 2.13: Texture Mapping with Hidden Surface Detection

Chapter 3

Feature Line Extraction

We obtained a textured 3D polygonal model by using the method described in the previous chapter. In this chapter, we propose a method to extract feature lines based on a virtual viewpoint selected arbitrarily.

First, we render the 3D model by the standard technique of 3D computer graphics using OpenGL. Here, a virtual viewpoint can be given easily by OpenGL's functions of perspective transformation. We employed OpenGL because of powerful and various functions that allow rapid development of a high-quality 3D graphic program, such as 3D coordinate system as default, automatic z-buffer calculation, and easy texture mapping. Feature lines are extracted on the basis of the rendered result and the virtual position of the viewpoint.

We propose a method to extract geometry and photometry edges in the following sections. As mentioned in Section 1.2, a textured 3D object provides not only photometry edges but also geometry edges, and geometry edges complement where few photometry edges are extracted. By combining these edges, the object is represented more precisely in 2D output.

3.1 Geometry Edges

Seen from a virtual viewpoint, we recognize an edge of a polygon as a geometry edge when the position of the edge is maximally away in a curved surface. Then normals of the edges should be nearly vertical to gaze direction. Based on this notion, our system judges if a vertex composes a geometry edge by using normals of each vertex.

We compute geometry edges as a set of vertices. The algorithm is as follows:

1. Calculate a normal vector n_i of each vertex v_i by interpolation from adjacent vertices and normalize it.

2. Represent a gaze direction as a direction vector p.

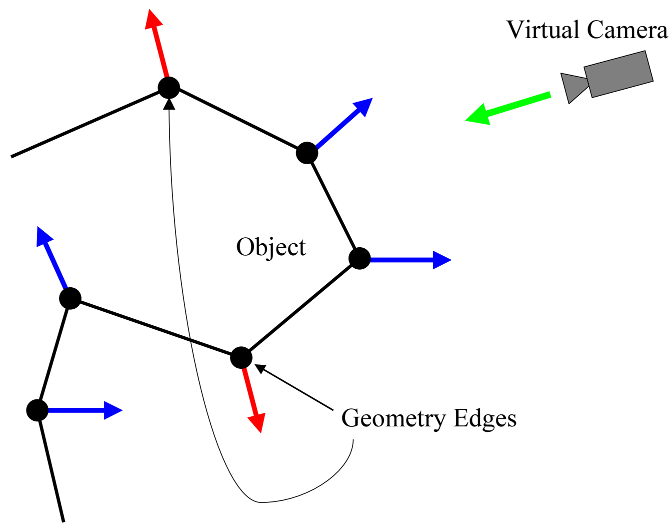


Figure 3.1: Model of Geometry Edges

3. Calculate an inner product $u_i = n_i \cdot p$ for each n_i.

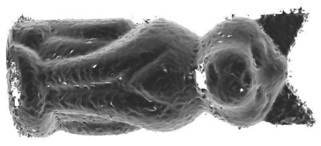
4. Compare u_i with predefined threshold$\tau > 0$. If $|u_i| < \tau$, v_i is a geometry edge.

This algorithm almost returns a sufficient number of vertices, but it fails where a curvature is abrupt and halfway vertices are skipped. To compensate for this defect, we added every vertex that satisfies one of following conditions to geometry edges by considering the xy coordinate of the vertex in the rendered result:

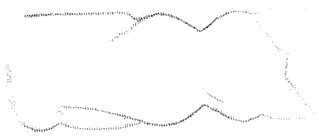
- x is the maximum of all vertices that have same y.
- x is the minimum of all vertices that have same y.
- y is the maximum of all vertices that have same x.
- y is the minimum of all vertices that have same x.

However, simple application of this algorithm causes some invalid edges if the polygon is sparse. For that reason we selected proper edges out of neighboring n candidates; if candidates $v_1, v_2, ..., v_n$ are calculated due to maximum x values for $y, y+1, ..., y+(n-1)$, only one candidate has the maximum x value of the n vertices. In our system, n is set to 3 based on the experimental result (see Figure 3.2).

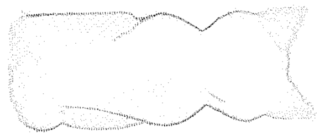
(a) Object



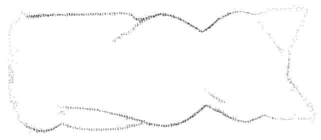
(b) Polygon



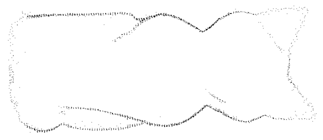
(c) Simple Edges



(d) Improved One $(n = 1)$



(e) Improved One $(n = 3)$



(f) Improved One $(n = 5)$

Figure 3.2: Result of Geometry Edge Extraction

3.2 Photometry Edges

Photometry edges are extracted from a 2D image of a rendered 3D model by applying a traditional 2D edge detection algorithm. In computer vision, edge detection from a 2D image is an old problem, and many solution have been offered mainly based on a derivative technique [28]. Among them, Canny's edge detector [29] is widely used as the current standard edge detection scheme because of high-quality edge detection; it can pick up delicate edges that cannot be found by another method.

This method treats edge detection as a signal-processing problem and specifies an objective function to be optimized. The objective function was designed to achieve the following optimization constraints:

- Important edges should not be missed, and there should not be no spurious response.
- The distance between the actual and located position of the edge should be minimal.
- Multiple responses to single edges should be minimized.

A first derivative of a Gaussian approximates this objective function very well.

Let $I(x,y)$ denote an image. The algorithm of this method is as follows:

1. Smooth the image with a Gaussian filter to reduce noise and unwanted details and textures.
$$g(x,y) = G_\sigma(x,y) \otimes I(x,y) \tag{3.1}$$
where
$$G_\sigma(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.2}$$
2. Calculate gradient of $g(x,y)$ using any of the gradient operators (Sobel, Prewitt, Roberts, etc.) to obtain
$$M(x,y) = \sqrt{{g_x}^2(x,y) + {g_y}^2(x,y)} \tag{3.3}$$
and
$$\theta(x,y) = \arctan\left\{\frac{g_y(x,y)}{g_x(x,y)}\right\} \tag{3.4}$$
3. Threshold M:
$$M_T(x,y) = \begin{cases} M(x,y) & if \quad M(x,y) > T \\ 0 & otherwise \end{cases} \tag{3.5}$$
where T is so chosen that all edge elements are kept while most of the noise is suppressed.

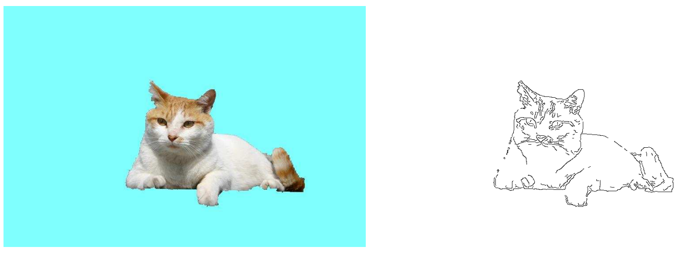
4. Since the edges might have been broadened in step 1, suppress non-maximal pixels in the edges in M_T obtained above to thin the edge ridges. In order to do so, check to see whether each non-zero M_T is greater than its two neighbor along the gradient direction $\theta(x,y)$. If not so, set it to 0.

5. Threshold the previous result by two different thresholds τ_1 and τ_2 (where $\tau_1 < \tau_2$) to obtain two binary images B_1 and B_2. Note that compared to B_1, B_2 has less noise and fewer false edges but larger gaps between edge segments.

6. Link edge segments in B_2 to form continuous edges. In order to do so, trace each segment in B_2 to its end and then search its neighbor in B_1 to find any edge segment in B_1 to bridge the gap until reaching another edge segment in B_2.

In our system, the parameters $\sigma, T, \tau_1, \tau_2$, which appeared in the algorithm, can be adjusted while seeing the result. In addition, we can designate how far to search connecting edges in the same way.

As seen in Figure 3.3, this method successes in extracting detailed feature lines. On the other hand, it fails in finding some important lines (e.g. the boundary between the face and the body) because of color similarity around the desirable line. Geometry edges are expected to complement these lines.



(a) Original Image (b) Photometry Edges

Figure 3.3: Result of Photometry Edge Extraction

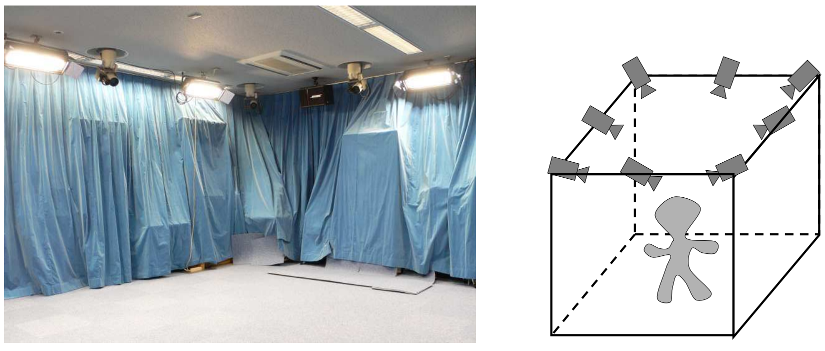
Chapter 4

Experimental Results

We implemented the system on the basis of the scheme described in Chapter 2 and 3, and applied it to pictures of a human taken with calibrated multiple cameras. In this chapter, we show our environment for experiment and a result obtained by means of our system.

4.1 Environment

We captured images of a human with 8 synchronized cameras arranged around the ceiling of the experiment room, as seen in Figure 4.1. These cameras are connected to clusters, and the clusters synchronize behavior of each camera and reserve pictures taken by each of them at the rate of 30 frames per second. These cameras are well calibrated in advance.



(a) Experiment Space

(b) Ceiling Cameras

Figure 4.1: Environment

We obtained 50 background pictures in which no humans are included, 50 foreground pictures in which a human is included, and calibration data from each camera described above. We utilized all of the background pictures for background subtraction described in Section 2.2.1, and one set of the foreground pictures that are captured at the same time.

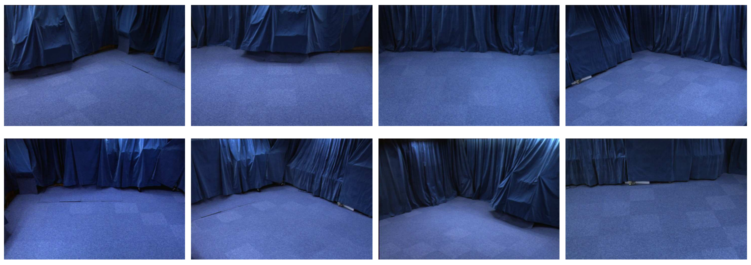


Figure 4.2: Background Images

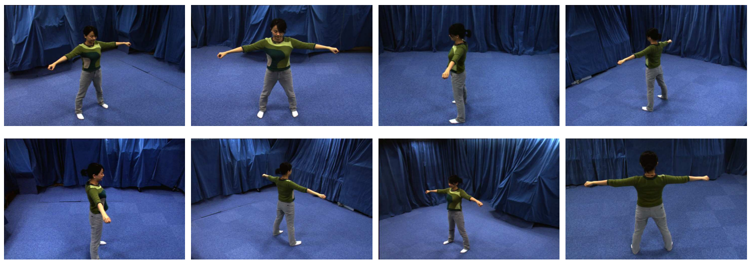


Figure 4.3: Foreground Images

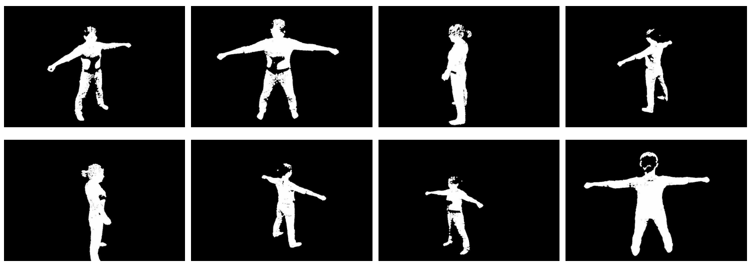
4.2 3D Shape Reconstruction

First, we constructed a textured 3D polygonal model of the human by using the technique described in Chapter 2.

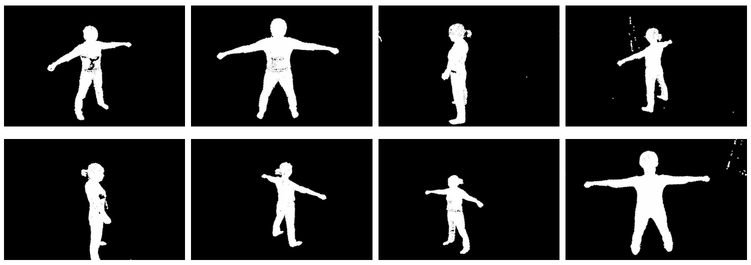
4.2.1 Silhouettes

As the result of silhouette extraction proposed in Section 2.2, we obtained three results shown in Figure 4.4: results of background subtraction for two detection rate r_1, r_2 and the final resultant silhouette by means of graph-cut algorithm. Based on the result of preliminary experiments, r_1 and r_2 are set to 0.999 and 0.7 respectively.

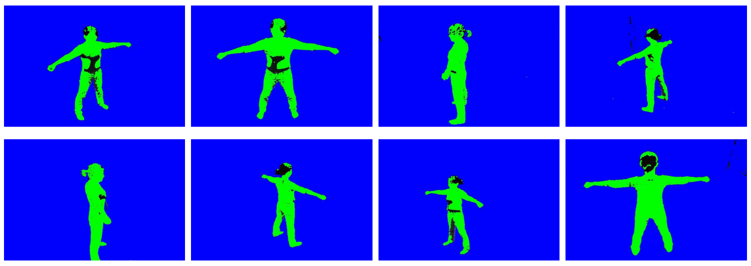
We found in each result of Figure 4.4(a) that no foreground regions are outside the human and some important inside regions are separated from largest foreground region, and therefore we did not utilize the function to eliminate invalid foreground regions. On the other hand, in



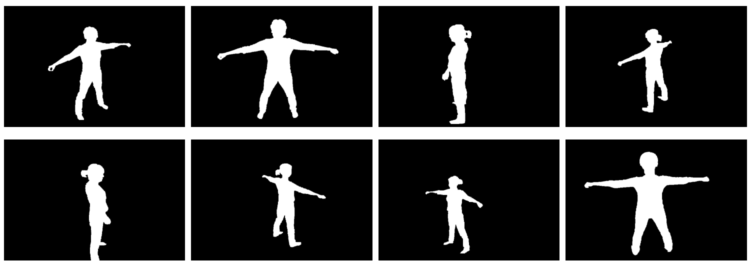
(a) Background Subtraction (detection rate $r_1 = 0.999$)



(b) Background Subtraction (detection rate $r_2 = 0.7$)



(c) Designation



(d) Final Silhouette

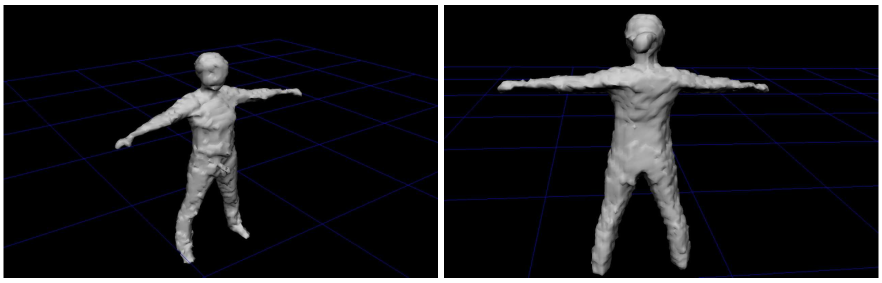
Figure 4.4: Result of Silhouette Extraction

each result of Figure 4.4(b) some background regions are inside the human and therefore we utilized the function to eliminate invalid background region.

Figure 4.4(d) indicates good separations of foreground and background region, and suggests that our method is effective for refining the separation between the foreground and the background region in the captured images. However, our experimental dataset is well separated only by background subtraction to some extent, and we should examine the effectiveness of our method by a variety of environments.

4.2.2 Textured 3D Polygon

We reconstructed the 3D polygonal model from obtained silhouettes of the human, by applying the methods described in Section 2.1 and 2.3. Then we texture on the model using the technique proposed in Section 2.4. The resultant polygonal model and textured model is shown in Figure 4.5 and 4.6.



(a) (b)

Figure 4.5: 3D Polygonal Model

These results are sufficient to represent whole shape of the human body, and textured well without wrong correspondence such as seen in Figure 2.13. However the surface of the model is rough and some background textures are mapped on it. The quality of the resultant model can be improved by integrating photo consistency with our method, or by introducing some smoothing method to the result.

4.3 Line Extraction

We rendered the 3D model obtained in previous section from an arbitrary virtual viewpoint, as we saw in the previous section. Then two types of feature lines are extracted and

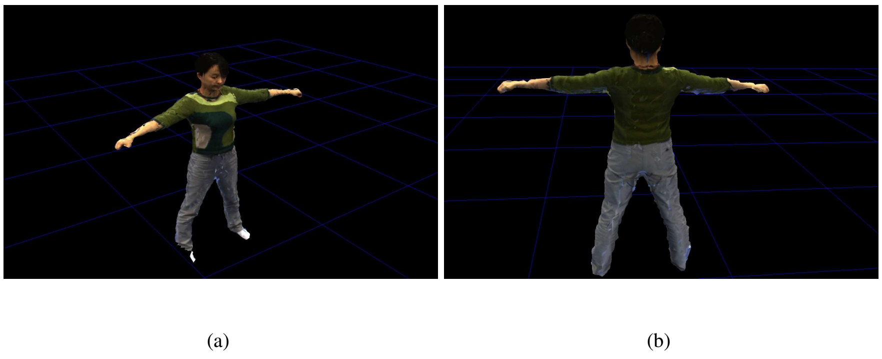

(a) (b)

Figure 4.6: Textured 3D Model

combined.

4.3.1 Geometry Edges

We obtained geometry edges shown in Figure 4.7 from the obtained polygonal model by using the method described in Section 3.1.

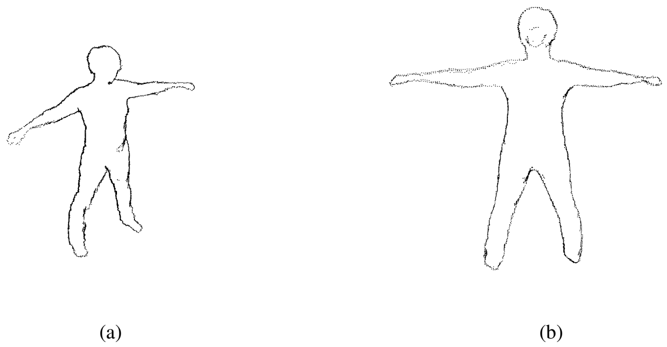

(a) (b)

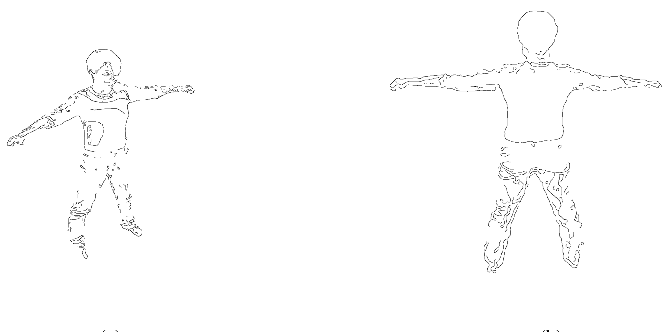
Figure 4.7: Geometry Edges

In this experiment we did not utilize a dataset which has a characteristic shape, and therefore we can confirm only a few merits of geometry edges. We can see that geometry edges represent silhouette of the human well in both of the Figure 4.7, and the hairstyle of the human in Figure 4.7(b). We cannot recognize the hairstyle of the human from Figure 4.6(b) and this is the typical case where geometry edges contributes to recognition of the object.

At present, geometry edges are expressed by vertices and a robot cannot draw them as it is. We should convert them to a line expression using connection of the vertices.

4.3.2 Photometry Edges

We obtained photometry edges shown in Figure 4.8 from the rendered result of textured 3D model by using the method described in Section 3.2.



(a) (b)

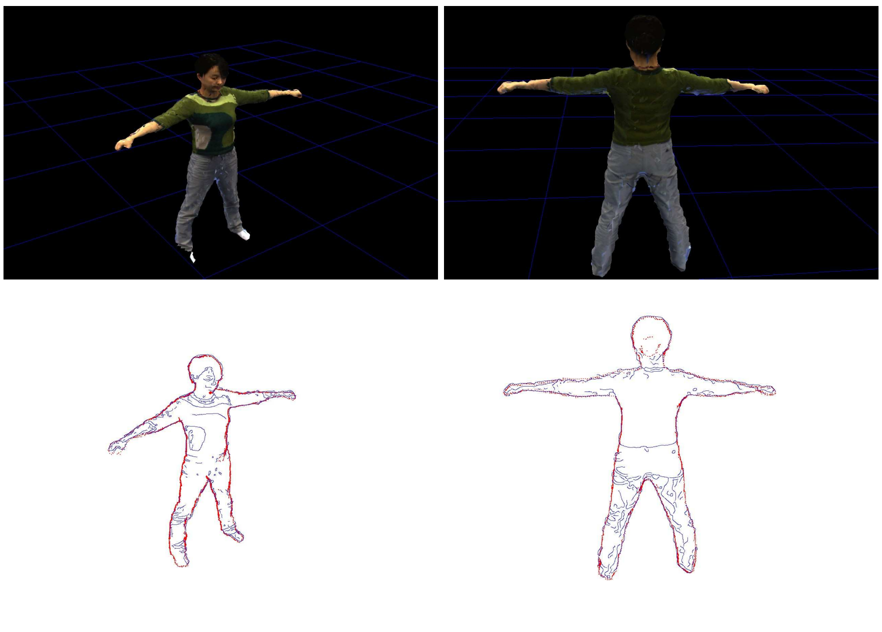
Figure 4.8: Photometry Edges

The resultant edges offer some features of the model, such as boundaries and patterns of clothes. However they contain some noises and are not enough to represent the human. Although this is because of insufficient quality of the 3D model in part, we should contrive a method to remove noises and refine the result.

4.3.3 Combined Edges

We combined two types of edges obtained in the previous parts of this section. The result is shown in Figure 4.9.

The result indicates that both two types of edges complement each other. In Figure 4.9(b), the hairstyle of the human is represented only by geometry edges, and the boundary of pants. The quality of the result progresses by improvement of the quality of each type of edges.



(a) (b)

Red and blue areas denote geometry and photometry edges respectively.

Figure 4.9: Final Resultant Edges

Chapter 5

Conclusion

5.1 Summary

We proposed a feature extraction system that yields desirable lines to be drawn by a painter robot from images obtained from multiple cameras. This is the first step to realize painting by humanoid robot, and finally to understand human process of painting. Although we have some problems on the quality of results, we obtained some important achievements.

First, we implemented a system to automatically extract feature lines of the object. We can obtain a textured 3D polygonal model only by providing our system with captured images and camera calibrations, and feature lines only by deciding a virtual viewpoint. We cannot say that the extracted feature lines have sufficient quality. However we can conclude that they roughly represent the target object, and desirable feature lines can be obtained by improving quality as an extension of this work.

Second, we proposed a combination of background subtraction and graph-cut algorithm as a part of 3D modeling. This approach gives robustness to former background subtraction methods and enables us to obtain clear silhouette of the object. This technique can be applied not only to 3D shape reconstruction but also to other areas in computer vision, such as object recognition. However, we experimented this method for datasets to which background subtraction gives good result to some extent, and the effectiveness of this method should be examined more carefully.

5.2 Future Work

For the further improvement of our system, we are planning to work on the following tasks:

- Quality of feature line extraction

At present, we obtain both types of feature lines as a set of pixels and the quality of the lines are still insufficient. We will develop a method to convert the set of pixels to lines, and to refine and simplify the lines.

- Refinement of the 3D model
  Volume intersection has a limitation, such as reconstruction of concave parts. This problem can be solved by using detailed information of silhouettes, such as colors. We are developing a system that combines volume intersection and voxel coloring.

- Extension of our robust silhouette segmentation
  Our silhouette extraction technique can be extended to sequence photographs. We are expecting that a moving object can be segmented from background by using proposed method, and this approach will be useful in the field of motion tracking, intelligent transport systems(ITS), and so on.

References

[1] F. Nakajima, D. Ohkawara and A. Nakata: Portrait Drawing Robots, Journal of the Robotics Society of Japan, Vol. 3, No. 4, pp. 123-124, August 1985.

[2] S. Calinon and J. Epiney and A. Billard: A Humanoid Robot Drawing Human Portraits, In Proceedings of the IEEE-RAS International Conference on Humanoid Robots, pp.161-166, 2005.

[3] H. Tsuge, K. Tani and T. Tanaka: Study of Sensing Methods for Sophisticated Robot Teaching, Gifu Prefectural Research Institute of Manufacturing Information Technology, Gifu, Japan, Techinical Report, 2001.

[4] M. Stein and C. Madden: The pumapaint project: Long term usage trends and the move to three dimensions, In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), April 2005.

[5] T. Morita, J. Takamatsu, K. Ogawara, H. Kimura and K. Ikeuchi: Knot Planning from Observation, In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), pp.3887-3892, Sep. 2003.

[6] J. Takamatsu, K. Ogawara, H. Kimura and K. Ikeuchi: Understanding of Human Assembly Tasks for Robot Execution — Generation of Optimal Trajectories Based on Transitions of Contact Relations —, Journal of the Robotics Society of Japan, Vol. 22, No. 6, 2004.

[7] Danijela Markovic and Margrit Gelautz: Drawing the real, Proceedings of 3rd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia (GRAPHITE'05), pp. 237-243, 2005.

[8] Tong-Yee Lee, Shaur-Uei Ya, Yong-Nien Chen and Ming-Te Chi: Real-Time 3D Artistic Rendering System, Proceedings of Knowledge-Based Intelligent Information and Engineering Systems: 9th International Conference (KES'05), Vol. 3683, pp. 456-462, September 2005.

[9] Atsushi Kasao and Kazunori Miyata: Algorithmic Painter: a NPR method to generate various styles of painting, Journal of The Visual Computer, Vol. 22, No. 1, pp. 14-27, January 2006.

[10] Hajime Matsui, Henry Johan and Tomoyuki Nishita: A Method to Create Colored Pencil Style Images by Drawing Strokes Based on Boundaries of Regions, Proceedings of Computer Graphics International (CGI'05), pp. 148-154, June 2005.

[11] Yin Li, Jian Sun, Chi-Keung Tang, Heung-Yeung Shum: Lazy Snapping, ACM Transactions on Graphics, vol. 23(3), pp.303-308, August 2004.

[12] D. Marr and T. Poggio: Cooperative Computation of Stereo Disparity, Science, Vol. 194, No. 4262, pp. 283-287, Oct. 1976.

[13] Y.Yang, A. Yuille, and J. Lu: Local, Global, and Multilevel Stereo Matching, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 274-279, 1993.

[14] O. Faugeras and R. Keriven: Variational Principles, Surface Evolution, PDE's, Level Set Methods, and the Stereo Problem, IEEE Transactions on Image Processing, Vol. 7, No. 3, pp. 336-344, March 1998.

[15] W. Martin and J. K. Aggarwal: Volumetric Descriptions of Objects from Multiple Views, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 5, No. 2, pp.150-158, March 1983.

[16] A. Laurentini: The Visual Hull Concept for Silhouette-Based Image Understanding, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 2, Feb. 1994.

[17] S. Srivastava and N. Ahuja: Octree Generation from Object Silhouettes in Perspective Views, Computer Vision, Graphics and Image Processing, Vol. 49, No. 1, Jan. 1990, pp. 68-84.

[18] S. Seitz and C. Dyer: Photorealistic Scene Reconstruction by Voxel Coloring: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 1067-1073, June 1997.

[19] X. Wu, T.Wada, S.Tokai, T.Matsuyama: Parallel Volume Intersection Based on Plane-to-Plane Projection, IPSJ Transactions on Computer Vision and Image Media, Vol.42, No.SIG 6(CVIM 2), pp.33-43, June 2001.

[20] K. N. Kutulakos and Steven Seitz: A Theory of Shape by Space Carving, Technical Report TR692, Computer Science Dept., U. Rochester, May 1998.

[21] D. Harwood T. Horprasert and L.S. Davis: A statistical approach for real-time robust background subtraction and shadow detection, Proceedings of Asian Conference on Computer Vision, Jan 2000.

[22] S. Vincent and P. Soille: Watersheds in Digital Spaces: An Efficient Algorithm Based on Immersion Simulations, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 13, No. 6, pp.583-598, June 1991.

[23] S. Geman and D. Geman: Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 6, No. 6, pp. 721-741, 1984.

[24] Yuri Boykov and Marie-Pierre Jolly: Interactive Graph Cuts for Optimal Boundary & Region Segmentation of Objects in N-D images, In International Conference on Computer Vision(ICCV), Vol. 1, pp. 105-112, 2001.

[25] William E. Lorensen and Harvey E. Cline: Marching Cubes: A high resolution 3D surface construction algorithm, Proceedings of the 14th Annual Conference on Computer Grphics and Interactive Techniques, pp. 163-169, 1987.

[26] Alex Goriachev: Marching Cubes, Online Document, May 2002, Available at http://cs.technion.ac.il/~u_shani/cs236807-S2/

[27] Montani C. and Scateni R. and Scopigno R: Discretized Marching Cubes, Proceedings of IEEE Computer Society Visualization '94, pp. 281-287, 1994.

[28] D. Marr and E. C. Hildreth: Theory of edge detection, Proceedings of the Royal Society of London B, Vol. 207, pp. 187-217, 1980.

[29] J. Canny: A computational approach to edge detection, IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, pp.679-698, 1986.