ABSTRACT

2D AND 3D SURFACE IMAGE PROCESSING ALGORITHMS AND THEIR APPLICATIONS

by
Jianlin Gao

This doctoral dissertation work aims to develop algorithms for 2D image segmentation application of solar filament disappearance detection, 3D mesh simplification, and 3D image warping in pre-surgery simulation. Filament area detection in solar images is an image segmentation problem. A thresholding and region growing combined method is proposed and applied in this application. Based on the filament area detection results, filament disappearances are reported in real time. The solar images in 1999 are processed with this proposed system and three statistical results of filaments are presented.

3D images can be obtained by passive and active range sensing. An image registration process finds the transformation between each pair of range views. To model an object, a common reference frame in which all views can be transformed must be defined. After the registration, the range views should be integrated into a non-redundant model. Optimization is necessary to obtain a complete 3D model. One single surface representation can better fit to the data. It may be further simplified for rendering, storing and transmitting efficiently, or the representation can be converted to some other formats.

This work proposes an efficient algorithm for solving the mesh simplification problem, approximating an arbitrary mesh by a simplified mesh. The algorithm uses Root Mean Square distance error metric to decide the facet curvature. Two vertices of one edge and the surrounding vertices decide the average plane. The simplification results are

excellent and the computation speed is fast. The algorithm is compared with six other major simplification algorithms.

Image morphing is used for all methods that gradually and continuously deform a source image into a target image, while producing the in-between models. Image warping is a continuous deformation of a graphical object. A morphing process is usually composed of warping and interpolation. This work develops a direct-manipulation-of-free-form-deformation-based method and application for pre-surgical planning. The developed user interface provides a friendly interactive tool in the plastic surgery. Nose augmentation surgery is presented as an example. Displacement vector and lattices resulting in different resolution are used to obtain various deformation results. During the deformation, the volume change of the model is also considered based on a simplified skin-muscle model.

2D AND 3D SURFACE IMAGE PROCESSING ALGORITHMS AND THEIR APPLICATIONS

by
Jianlin Gao

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Electrical Engineering

Department of Electrical and Computer Engineering

May 2001

APPROVAL PAGE

2D AND 3D SURFACE IMAGE PROCESSING ALGORITHMS AND THEIR APPLICATIONS

Jianlin Gao

Dr. Mengchu Zhou, Dissertation Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Haimin Wang, Dissertation Co-Advisor — Date
Professor of Physics, NJIT

Dr. John D. Carpinelli, Committee Member — Date
Associate Professor of ECE & CIS, NJIT

Dr. Jason Geng, Committee Member — Date
President of Genex Technologies, Inc.

Dr. Yunqing Shi, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Jianlin Gao

Degree: Doctor of Philosophy

Date: May 2001

Undergraduate and Graduate Education:

- Doctor of Philosophy in Electrical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2001
- Master of Science in Automatic Control,
  Beijing Institute of Technology, Beijing, P.R.China, 1995
- Bachelor of Science in Automatic Control,
  Beijing Institute of Technology, Beijing, P.R.China, 1992

Major: Electrical Engineering

Presentations and Publications:

Jianlin Gao, Mengchu Zhou and Haimin Wang,
"Mesh simplification with average planes for 3-D image," *Proceeding of 2000 IEEE International Conference on Systems, Man & Cybernetics*, pp. 1412-1417, Nashville, TN, USA, October 2000.

Jianlin Gao, Mengchu Zhou and Haimin Wang,
"A threshold and region growing combined method for filament disappearance detection in solar images," *Proceedings of The 35th Annual Conference on Information Sciences and Systems*, Baltimore, Maryland, USA, March 2001.

Jianlin Gao, Mengchu Zhou and Haimin Wang,
"2D and 3D medical image database design," *2001 IEEE International Conference on Systems, Man & Cybernetics*, Tucson, Arizona, USA, 2001, Accepted.

Jianlin Gao, Mengchu Zhou, Haimin Wang and Congzhe Zhang,
"Three dimensional surface warping for plastic surgery planning,"
2001 IEEE International Conference on Systems, Man & Cybernetics, Tucson, Arizona, USA, 2001, Accepted.

Jianlin Gao, Haimin Wang and Mengchu Zhou,
"Development of an automatic filament disappearance detection system,"
Submitted to *Solar Physics* for review, 2001,

Congzhe Zhang, Mengchu Zhou and Jianlin Gao,
"Conversion between discrete images and organized 3D file format,"
2001 IEEE International Conference on Systems, Man & Cybernetics, Tucson, Arizona, USA, 2001, Accepted.

To my beloved family

ACKNOWLEDGMENT

I wish to express my sincere appreciation to my advisor, Dr. Mengchu Zhou and co-advisor, Dr. Haimin Wang for their help and guidance in pursuing this research. Their assistance throughout my years of study here has been invaluable. Special thanks are given to Dr. John D. Carpinelli, Dr. Jason Geng and Dr. Yunqing Shi for their active participation in my committee.

I would also like to thank all those who have contributed to this work by providing both data and results. Genex Technologies, Inc. provides facial model, ChristmasMan model. Dr. Jason Geng, Prof. Jianping Wang and Dr. Jinfa Xu at Genex Technologies gave me great help during my 3D research work. The bunny model is distributed publicly by the Stanford Graphics Lab. Hugues Hoppe at Microsoft provided the fandisk model. Roberto Scopigno gave me the same fandisk and bunny model as used in his work and the performance data of major mesh simplification algorithms.

I am grateful to the support of NSF, ONR and NASA through the Center for Solar Research at NJIT and Genex Technologies, Inc.

Many of my fellow graduate students in Electrical and Computer Engineering Department and Center for Solar Research are deserving of recognition for their support. Congzhe Zhang helped me greatly during the work of 3D warping application development. Tom Spirock carefully read the manuscript of Chapter 2. Ying Tang, Shu Yang and Jingshan Wang gave me many supports during my graduate research.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

Chapter **Page**

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

1.1 Motivation

The dissertation research is to develop a 2D image segmentation algorithm in solar images to detect filament disappearance, to develop 3D algorithms for mesh simplification and warping technique in plastic surgery planning applications. The study of solar filament is one of the very important subjects in solar physics. Filaments are amazing objects: they are located in corona but possess temperature a hundred times lower and densities a hundred times greater than the respective corona values. Filaments divide magnetic regions with opposite polarities. Their eruptions are associated with geomagnetic storms that may affect power system, communication and safety of space missions. Big Bear Solar Observatory generates 600 images per day. This work develops one automatic filament detection system with image processing techniques.

Three-dimensional surface reconstruction is an important intermediate goal of many vision systems. A useful surface reconstruction theory has applicability to a wide variety of fields, for example, design automation, reverse engineering, manufacturing automation, terrain mapping, vehicle guidance, archeology, restoration of arts, surveillance, and intelligent robots in general. Humans can communicate and understand a number of shapes almost effortlessly. However, finding a useful and general method for machine representation of shapes has proven to be a nontrivial problem. Methods that use numerical specifications are usually limited in generality. They have enormous amounts of precision but are commonly restricted to a specific domain [Bolle et al., 1991].

A surface description of an object is usually obtained in four steps. First, 3D information is acquired from a scene. There are two different approaches: passive and active. The second step is range image registration to obtain the relationship between different images of the same object. The third step is range image integration. After the relationships between different views are decided, a set of registered range images should be integrated into a non-redundant surface model. The fourth step is mesh surface simplification. Surface simplification is useful in order to make storage, transmission, computation, and display more efficient. This work studies the 3D surface reconstruction methods and proposes one surface simplification algorithm.

Image warping and morphing are important graphical operations, with applications in many areas, which are used to transform graphical objects. Warping deforms a single object. Morphing interpolates between two objects. These transformations can be applied to the various types of graphical objects, such as 2D drawings, images, surfaces, and volumes. There are now many breath-taking examples in film and television of the fluid transformation of one digital image into another. This work applies free-form deformation into the pre-surgery simulation. Plastic surgery is a routine procedure to correct congenital deformities or to treat the deformities caused by accidents. The motivation of free-form deformation for precise pre-surgical planning is to help surgeons reduce the potential of risks in surgery and save the test time. For example, in nose and breast augmentation, it is highly desired to test different implants using 3D models and graphical user interface before the surgery. Currently, the patients have to tolerate the hurt and time-consuming test with various size and form implants during a surgery process. The implant material factory has to produce three or four times of those that

patients actually need. A surgeon also has to order up to a dozen of different implants for one patient in order to find the most suitable one during the process. This work intends to provide one convenient method and computer tool to help simplify this process for surgeons. It assumes much significance since the results could potentially reduce the long time the patient has to suffer, improves a surgeon's productivity and decrease factory manufacturing cost, thereby reduce the overall surgery cost.

1.2 Objectives

The goal of this research is to develop useful image processing algorithms for applications in solar images, efficient transmission and storage of 3D images and 3D medical images in plastic surgery. The specific objectives are as follows:

(1) Image segmentation in solar filament disappearance application

Big Bear Solar Observatory (BBSO) generates a solar image around one minute. One has to develop an efficient filament detection algorithm. The algorithm must meet the requirements of computation speed and accuracy.

After the filaments in one single image are detected, the filament disappearances should be detected in real time by tracking the filament areas in different images. The images can be transferred from BBSO on one scheduled time daily. The program processes the images and compares the filament results in two consecutive images and sends the results out via email.

In order to verify the accuracy and efficiency of the program, it is necessary to test the program on one whole year's solar images and try to find out some valuable statistical results.

(2) 3D mesh simplification

The second objective of this dissertation is to develop 3D mesh simplification. This work aims to propose an efficient algorithm for solving the mesh simplification problem. The simplification begins with a geometric description of an object and produces a new description that is similar in appearance to the original and that has many fewer geometric primitives. The proposed algorithm needs to be compared with other major mesh simplification algorithms.

(3) 3D image warping in pre-surgery simulation

3D image warping and morphing describes the methods for deforming images to arbitrary shapes. The approaches are based on volume representation or boundary representation of the 3D objects.

This work aims to develop a method and application for pre-surgical planning with 3D image warping technique. Precise pre-surgical planning can help surgeons to reduce the potential of risks in plastic surgery. The user interface needs to provide a friendly interactive tool in the pre-surgery planning. Few simple parameters should be set to adjust the surgery area and overall result. During the deformation, the volume change is also considered in order to approximate the implanted material size.

1.3 Organization

The dissertation is composed of three parts as discussed above. Chapter 2 discusses the image segmentation algorithm application in 2D solar filament disappearance application. Chapter 3 introduces 3D surface reconstruction system, which includes data acquisition, range image registration, integration and optimization. Chapter 4 presents the mesh

simplification algorithm proposed in this work. Comparison results are also presented. Chapter 5 introduces the background and related works in 2D and 3D image warping and morphing. Chapter 6 presents the application developed in this work on 3D surface warping for pre-surgical planning. Nose augmentation is discussed as an application example. Chapter 7 summarizes the contributions of this dissertation, discusses the limitations and indicates the possible directions of future works.

CHAPTER 2

IMAGE SEGMENTATION IN 2-D SOLAR FILAMENT DISAPPEARANCE DETECTION

2.1 Introduction

Big Bear Lake in California is characterized by 300 sunny days a year. Big Bear Solar Observatory (BBSO) is located in the mountain-lake giving it an excellent observing condition. BBSO observes continuously from sunrise to sunset. Now with one recently installed 2k × 2k camera, BBSO can obtain full disk Hα images at a rate of one frame per second, which is unprecedented in the history of full-disk patrols. Taking these advantages, an automated early warning system of filament eruption can be established. Filaments divide magnetic regions with opposite polarities. Their eruptions are associated with geomagnetic storms that may affect a power system, communication and safety of space missions. A computer program was developed to automatically recognize the structure of filament. When a filament disappears, this news is immediately broadcast via E-mail to researchers who are interested in the project and on BEARALERT mailing list. The broadcast information includes the time of its eruption, and coordinate and size of a filament. Based on the list of filament eruptions recorded by the automated program, a filament index (as a function of time) is established. Slight modification of the program can lead to a similar automated early warning system to monitor onset of solar flares, another kind of violent solar magnetic storm.

The study of solar filament is one of a few very important subjects in solar physics—and it is closely related to the Space Weather project. Filaments are amazing objects: they are located in corona but possess temperature a hundred times lower and densities a hundred times greater than the respective corona values. In Hα images,

filaments are dark ribbons against bright disk. At the limb, they become bright features against sky, and are called prominence. Martin [1998] gives a comprehensive review of observational conditions for the formations and maintenance of filaments. Dumitrache [1997] describes the evolution of a filament.

2.2 Review of Image Segmentation

The filament area detection addresses the classical problem of image segmentation. Two factors are considered most important to make the detection meaningful. The first one is accuracy. The automatic method should detect the exact filament areas with small error. The second factor is speed since the solar image data is large and a number of images must be processed every day. Any disappearance of filament must be reported in real time.

Image segmentation is one of the most widely used steps in the extraction process of useful information from images. It can be defined as one process that partitions a digital image into disjoint (non-overlapping) regions. Many techniques are available to deal with the image segmentation problems. Haralick and Shapiro [1985], Pal and Pal [1993] and Castleman [1996] give excellent reviews on image segmentation techniques.

Image segmentation can be approached from three different philosophical perspectives. In the case of a region approach, one assigns each pixel to a particular object or region. In a boundary approach, one attempts only to locate the boundaries that exist between the regions. In an edge approach, one seeks to identify edge pixels and then link them together to form the required boundaries. All three approaches are useful for

visualizing the problem. Based on these three approaches, image segmentation methods can be broadly grouped into the following four categories.

2.2.1 Thresholding

Thresholding is a particularly useful region-approach technique for scene containing solid objects resting upon a contrasting background. It is computationally simple and never fails to define disjoint regions with closed, connected boundaries. Many thresholding techniques have been proposed [Sahoo et al., 1988; Weszka and Rosenfeld, 1978; Papamarkos and Gatos, 1994].

When using a threshold rule for image segmentation, all pixels are assigned at or above the threshold gray level to the object. All pixels with gray level below the threshold fall outside the object. The boundary is then that set of interior points, each of which has at least one neighbor outside the object.

However, this method works well only under strict conditions. If objects differ from their background by some property other than gray level (texture, etc.), one has to use an operation that converts that property to gray level. Then gray-level thresholding can segment the processed image [Castleman, 1996].

In the simplest implementation of boundary location by thresholding, the value of the threshold gray level is held constant throughout the image. If the background gray level is reasonably constant throughout, and if the objects all have approximately equal contrast above the background, then a fixed global threshold-based method usually works well, provided that the threshold gray level is properly selected.

The global thresholding method may not work on certain occasions, particularly when the image is noisy. The background gray level is not constant, and the contrast of objects varies within the image. In such cases, the objects may still be lighter or darker than the background, but any fixed threshold level for the entire image usually fails to separate the objects from the background. This leads one to the methods of adaptive thresholding [Castleman, 1996]. In adaptive thresholding normally an image is partitioned into several non-overlapping blocks of equal area and a threshold for each block is computed independently. These local thresholds are then interpolated over the entire image to yield a threshold surface.

2.2.2 Edge-Based Techniques

The preceding region approaches accomplish segmentation by partitioning the image into sets of interior and exterior points. By contrast, boundary approaches attempt to find the edges directly by their high gradient magnitudes. The image edges are first detected and then grouped into contour or surface that represents the boundaries of an image object. Usually the differentiation filter is used to get the first and second order image gradient. Then the candidate edges are extracted by thresholding the gradient [Henstock and Chelberg, 1996; Venkatesh and Kitchen, 1992; Kitchen and Rosenfeld, 1981; Torre and Poggio, 1986; Mclean and Jernigan, 1988; Nalwa and Binford, 1986]. Typically detected edges are incomplete, especially for noisy images, and linking of broken edges is a difficult procedure.

Edge detection techniques are classified into two categories: sequential and parallel. In the sequential technique the decision whether a pixel is an edge pixel or not is

dependent on the result of the detector at some previously examined pixels. On the other hand, in the parallel method the decision whether a point is an edge or not is made based on the point under consideration and some of its neighboring points. As a result of this, the operator can be applied to every point in the image simultaneously. The performance of a sequential edge detection method is dependent on the choice of an appropriate starting point and how the results of previous points influence the selection and result of the next point.

There are different types of parallel differential operators such as Roberts gradient, Sobel gradient, Prewitt gradient and the Laplacian operator. These difference operators respond to changes in gray level or average gray level. The gradient operators, not only respond to edges but also to isolated points. The Roberts operator marks edge points only. It does not return any information about the edge orientation. For Prewitt's operator the response to the diagonal edge is weak, while for Sobel's operator it is not that weak as it gives greater weights to points lying close to the point (x,y) under consideration. Both Prewitt's and Sobel's operators possess greater noise immunity. The preceding operators are called the first difference operator. Laplacian, on the other hand, is a second difference operator. The Laplacian operator is given by

$$\nabla^2 = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{2.1}$$

The digital Laplacian as a second difference operator, has a zero response to linear ramps. It responds strongly to corners, lines, and isolated points. Thus for a noisy picture, unless it has a low contrast, the noise produces higher Laplacian values than the edges. Moreover, the digital Laplacian is not orientation invariant. A good edge detector should be a filter with the following two features. First, it should be a differential operator,

taking either a first or second spatial derivative of the image. Second, it should be capable of being tuned to act at any desired scale, such that large filters can be used to detect blurry shadow edges, and small ones to detect sharply focused fine details. The second requirement is very useful as intensity changes occur at different scales in an image. The most satisfactory operator fulfilling these conditions is the Laplacian of Gaussian (LoG) operator. It is normally denoted by $\nabla^2 G$; where the Laplacian is as given by equation (2.1) and

$$G = e^{(x^2+y^2)/(2\pi\sigma^2)} \tag{2.2}$$

is a two-dimensional Gaussian distribution with standard deviation σ. The Gaussian part of the LoG operator blurs the image, wiping out all structures at scales much smaller than σ. The Gaussian blurring function is preferred over others because it has the desirable property of being smooth and localized in both spatial and frequency domains [Pal and Pal, 1993].

2.2.3 Region-Based Techniques

The goal is to detect regions (connected sets of pixels) that satisfy certain predefined homogeneity criteria. Two of the most widely used methods are split-and-merge and region growing [Russ, 1992]. Split-and-merge is a top-down method that begins with the entire image. In each step, each heterogeneous image region of the image is divided into four rectangular segments and the process is terminated when all regions are homogeneous. After the splitting stage a merging process is applied to unify the resulting similar neighboring regions [Chen et al., 1991; Haris et al., 1998].

Region growing [Besl, 1988; Beveridge, 1989] is an approach to image segmentation that has received considerable attention in the computer vision area of the artificial intelligence community. With this approach, one begins by dividing an image into many tiny regions. These initial regions may be small neighborhoods or even single pixels. In each region, suitably defined properties that reflect membership in an object are computed. The properties that distinguish the pixels inside the different objects might include average gray level, texture, or color information. Thus, the first step assigns to each region a set of parameters whose values reflect the object to which they belong.

Next, all boundaries between adjacent regions are examined. A measure of boundary strength is computed utilizing the differences of the averaged properties of the adjacent regions. A given boundary is strong if the properties differ significantly on either side of that boundary, and it is weak if they do not. Strong boundaries are allowed to stand, while weak boundaries are dissolved and the adjacent regions merged.

The process is iterated by alternately recomputing the object membership properties for the enlarged regions and then dissolving weak boundaries. The region-merging process is continued until a point is reached where no boundaries are weak enough to be dissolved. Then image segmentation is complete. Monitoring this procedure gives one the impression of regions in the interior of object growing until their boundaries correspond to the edges of the object.

Region-growing algorithms are computationally more demanded than the simpler techniques, but they are able to utilize several image properties directly and simultaneously in determining the final boundary location. Perhaps it shows the greatest

promise in the segmentation of natural scenes, where strong a priori knowledge is not available [Castleman, 1996].

2.2.4 Hybrid Techniques

Edges and regions are two complementary features of segmented images. Therefore, the hybrid methods combine both edge and region-based methods and are frequently employed for solving image segmentation problem [Haris et al., 1998; Ahuja, 1997; Chu and Aggarwal, 1993; Wani and Batchelor, 1994; Zhao and Zhang, 1997]. They represent an improved solution.

The algorithm in this work combines thresholding and region-based techniques. The filament area in a full-disk solar image is well separated from its background. The candidate filament pixels can be classified using thresholding. A region growing method can merge the candidate pixels to form filament areas.

2.3 Filament Detection System

BBSO produces a large number of Hα full-disk solar images (about 600 images/day). Denker et al. [1999] describe the instruments, the data calibration procedures, and image processing techniques used to obtain daily Hα full-disk observations. To process these images and find filament disappearances, it is essential to use an automated detection. This can bring us the following advantages: it is fast, accurate and cost can be reduced by a computer program. Wang et al. [1998] analyze the filament disappearances during the period of September 1991 through September 1994 and make two statistical studies. At that time, all the events were detected manually.

The essential component of the project is to develop a comprehensive program to detect and report filament disappearance as soon as it happens. Because people want to minimize the interference with the routine Hα observations, a separate PC (PC2) is used to run the program and to interface with the data acquisition PC (PC1). Every 10 minutes, a full disk Hα image is converted into JPEG file in PC1, and transferred to PC2. The main tasks of PC1 are to input original data, and acquire high-contrast full-disk images using image processing methods. These methods include centering images, computing a gain table and limb-darkening profile, and correcting images [Denker et al., 1999].

PC2 uses a corrected solar image to run the program. The central component of the program is to detect filaments accurately and fast. The program is developed in Interactive Data Language (IDL) from Research Systems, Inc. The program runs on Pentium II personal computer operating under Linux. The filament detection algorithm is discussed as follows.

Solar images are originally recorded by KODAK Megaplus Model 4.2 CCD camera that provides an array of 2032 (horizontal) × 2028 (vertical) pixels. After the processing of PC1, each image has 1920 × 1920 pixels. It is scaled to 1024 × 1024 in this work. The detection algorithm runs fast despite of large data array size (less than 1 minute for each image).

Because the filament areas are dark in nature, it is necessary to set a threshold value to select filaments. The median of intensity of the whole image is used as a reference since it is less sensitive to asymmetric intensity distributions than the mean value. The intensity threshold value is then set at 50% of the median value.

2.3.1 Filament Detection Algorithm

This work develops an algorithm that runs fast and detects filaments accurately. A flow chart of the algorithm is shown in Figure 2.1.

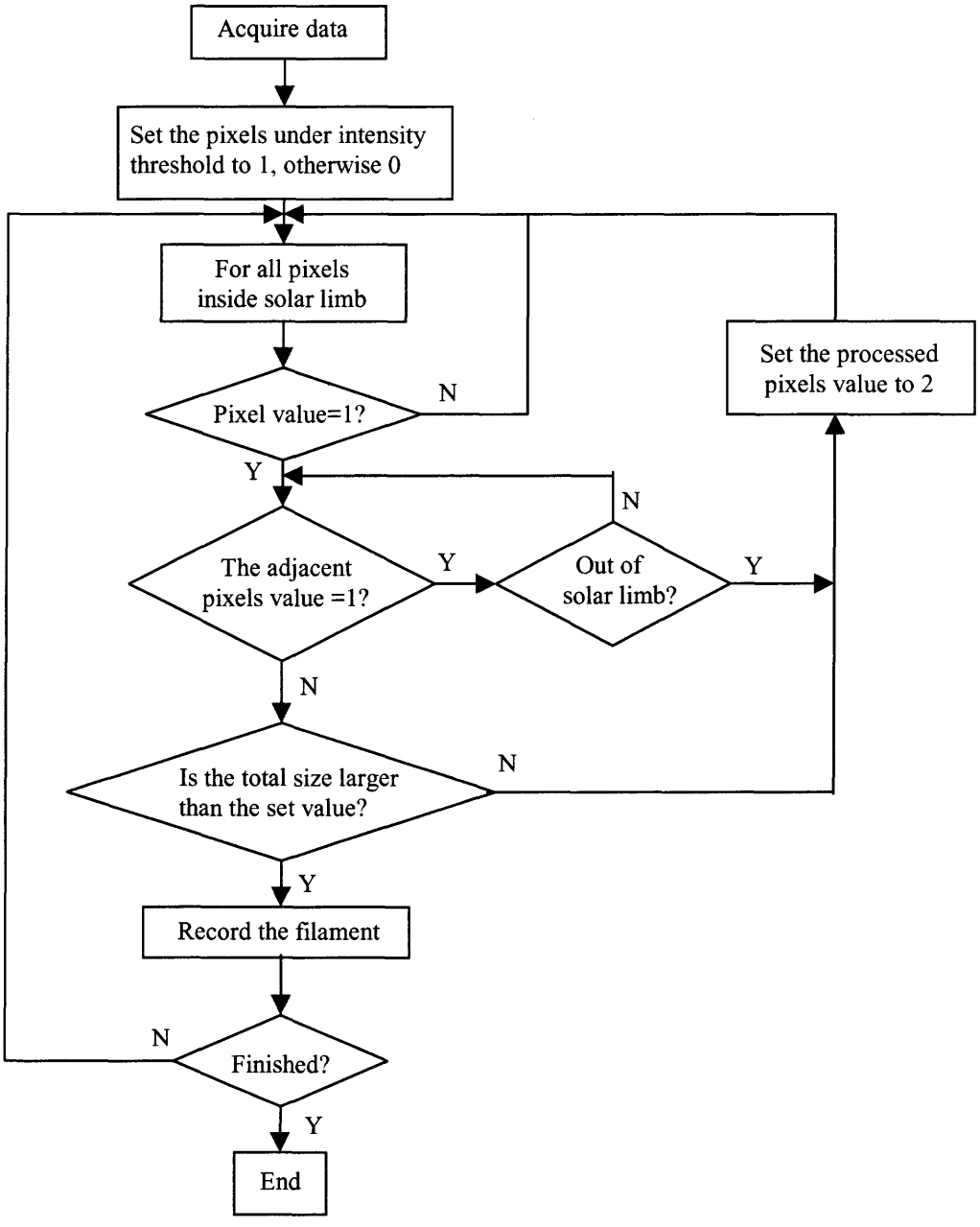


Figure 2.1 The flow chart of an efficient algorithm for filament detection.

The following four important issues should be considered in the algorithm:

(1) Thresholds

There are two different threshold values in the algorithm. The first threshold is used to detect the dark filament pixels by their intensity values. This threshold will be referred to as the "intensity threshold" throughout this chapter. Another threshold is used to determine if a detected dark area is large enough to be regarded as a filament by the total number of dark pixels in the region. This threshold will be referred to as the "size threshold" in this chapter.

(2) Dark areas outside the limb

In the process of detecting a filament, the adjacent pixels may be out of the solar limb. It is possible that a pixel outside the solar limb can be mistaken as part of a filament. In the program, one additional condition is set. If the filament area under consideration touches the solar limb the pixel will be deleted. This deletion does not affect the tracking disappearing filaments because people consider the filaments on the disk of the solar image more than those near the solar limb. When a filament appears beyond the solar limb, it appears as a bright structure known as a prominence. At this time, the developed program does not detect prominences.

(3) Adjacent pixel connection

A "4-pixel distance" method is used to connect the adjacent pixels in the detection of individual dark areas. The method is shown in Figure 2.2. In Figure 2.2(a), only the eight pixels that are immediately adjacent to the candidate pixel under consideration are checked during region growing. Because of the presence of error pixels, additional "adjacent" pixels

should be checked. Thus, so called "bridge pixels" must be added to allow such filaments to be correctly detected. The "4-pixel distance" method is applied to the data to repair the error pixels in the solar images. Figure 2.2(b) shows the "4-pixel distance" method. A total of eighty adjacent pixels to the candidate pixel are checked with this procedure.

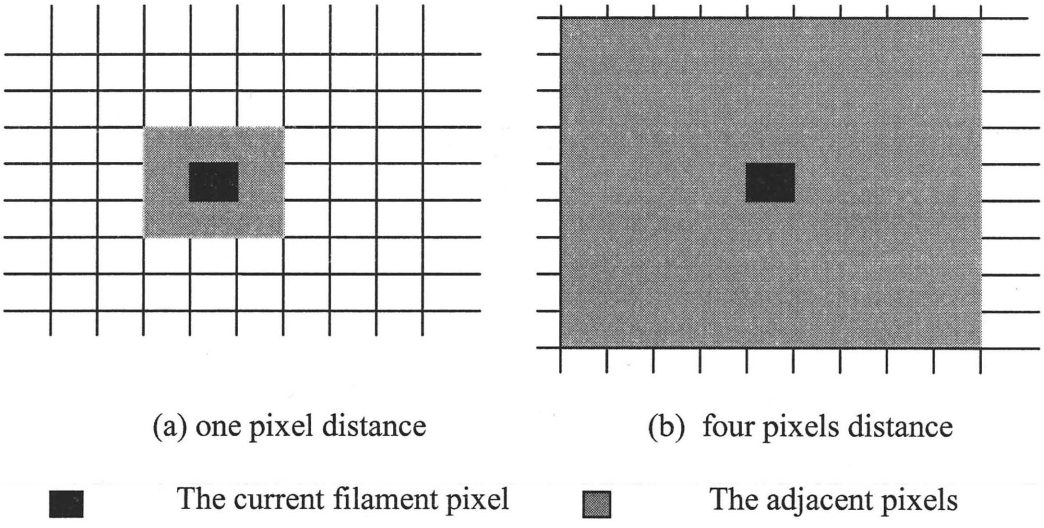


Figure 2.2 The adjacent pixels connection.

(4) In some solar images, a single large filament may be broken into several smaller filaments by error pixels. For example, the actual area of a filament may be 300 pixels. If it is separated into three smaller areas that have 150, 100 and 50 pixels respectively, the filament will be considered to be noise since all three areas are under the "size threshold" of 220 pixels. To ensure the program detects this area as one large filament, these error pixels must be detected and repaired. If the total distance between two adjacent filaments is less than 40 pixels, the two separately detected filaments are considered the same, larger filament. This method is referred to as the "40-pixel distance" method.

If the "40-pixel distance" method is independently applied to the solar images, many error filaments will be detected because of the many error pixels. Therefore, the "4-pixel distance" method is used to detect the individual small filaments and the "40-pixel distance" method is used to determine if two or more adjacent filaments are actually one large filament. Unfortunately, the "40-pixel distance" method runs very slowly and would take longer to process a single image than requirements demand. Therefore, the program is most efficient when both the "4-pixel distance" and "40-pixel distance" methods are used together.

The "intensity threshold" value is selected to be a half of the median intensity value of a solar image. Any pixel whose intensity value is over this "intensity threshold" value is set to 1. Otherwise, the pixel value is set to zero. The algorithm begins with one reference pixel whose intensity is under the "intensity threshold". Each adjacent pixel is then checked. If at least one adjacent pixel is below the "intensity threshold", the pixel is added to the region and is regarded as the current reference pixel. The adjacent pixels of the new reference pixel are then checked. The process continues until the area's adjacent pixels are all above the "intensity threshold". The total number of pixels in the area is then obtained. If the total exceeds the "size threshold" value, it is considered a filament. The pixel intensity value of this area is then set to 255. If the total number of pixels in the area under consideration is below the "size threshold", the area is considered to be not a filament and its pixel intensity values are set to 2. As the program continues to check for additional filaments on the solar disk, these pixels are no longer tested. The actual intensity and "size threshold" levels for the program are determined by the tests on the BBSO data.

2.3.2 Filament Disappearance Detection

After the filaments are detected in individual images, the results from two consecutive images are compared to detect filament disappearances. Taking into account the solar rotation of the sun during the time between the two images under consideration, the location of filament disappearances can be detected. The first implementation is to select one image per day to detect filament disappearances during that particular two days.

The following issues are important in the process of filament disappearance detection.

(1) Solar rotation

Solar rotation can carry a filament over the west limb of the sun and out of the view without any physical changes in the filament's structure. Therefore, this program must take this effect into account and not log a filament that rotates out of view as a disappearing filament.

(2) Position of the sun on the detector

Due to slight errors in telescope tracking, the position of the image of the sun on the detector can change by a small amount. If not taken into account, this effect can mimic a filament disappearance.

It is very important that the solar community be alerted to filament disappearances in a timely manner. To accomplish this task, the program periodically downloads all recent full-disk Hα images, detects all filaments, categorizes them into new, partially disappeared, fully disappeared, stable and growing and alerts the solar community, via email, should any relevant events be detected.

2.4 Results

2.4.1 Filaments Detected

Figure 2.3 presents one example of filament detection and filament disappearance detection. To ensure that the smallest filaments will be detected, the "size threshold" of the filament areas is set to 205 arcsec^2. While the "size threshold" of disappearing filament is set to 750 arcsec^2. The filament disappearances are reported in Table 2.1.

Table 2.1 Filament disappearances between Feb. 22, 1999 and Feb. 23, 1999.

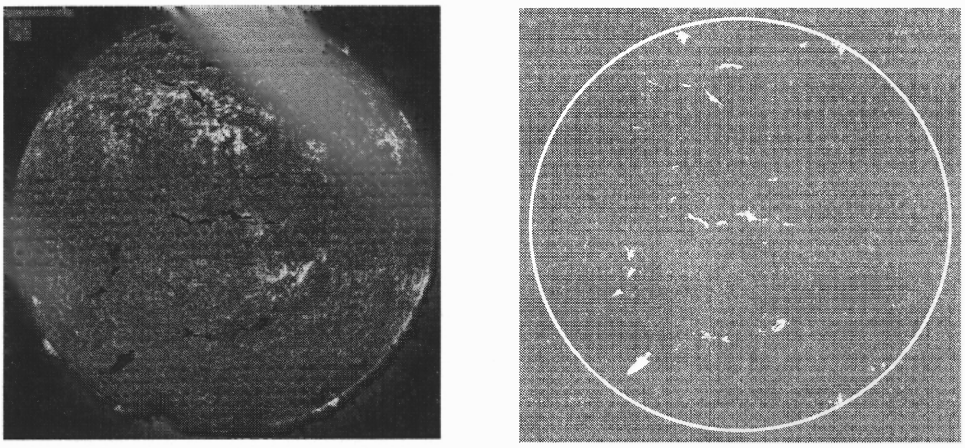
Event Number	Position	Area (Arcsec^2)
1	S13 E32	1211
2	S05 E11	1318
3	N30 E08	1184
4	S04 W02	2009

All of the available solar images from 1999 have been analyzed for this work. Three statistical studies of filaments and detected filaments disappearing, (1) size distribution (2) filament frequency in time series and (3) latitude distribution, based on this set of data are presented. The intention was not to present new results but to check that the filament detection algorithm is working correctly. The three statistical results are discussed below.

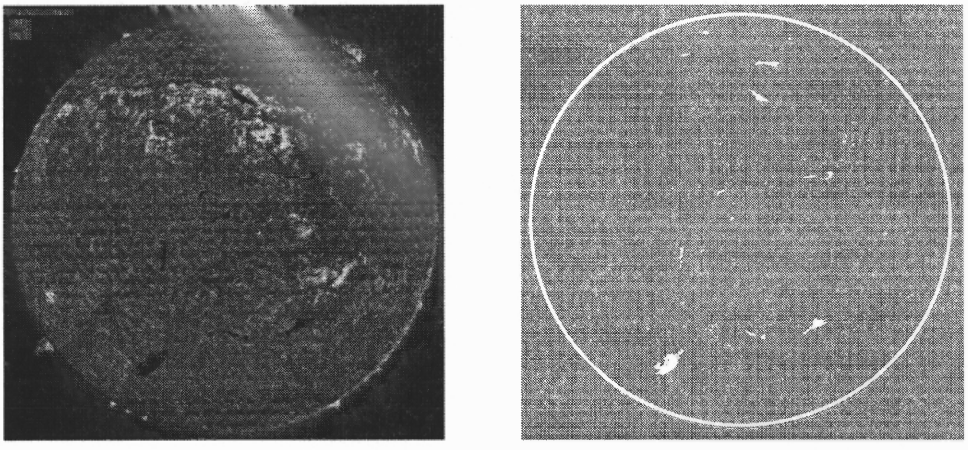
2.4.2 Size Distribution of Filament Disappearance Areas

In Figure 2.4, the size distribution of all the detected filaments and filament disappearances are shown. The filament "size threshold" is set at 750 Arcsec^2. Using a curve fitting technique, the relationship between size distribution (N) and the filament disappearance area (S) is as follows:

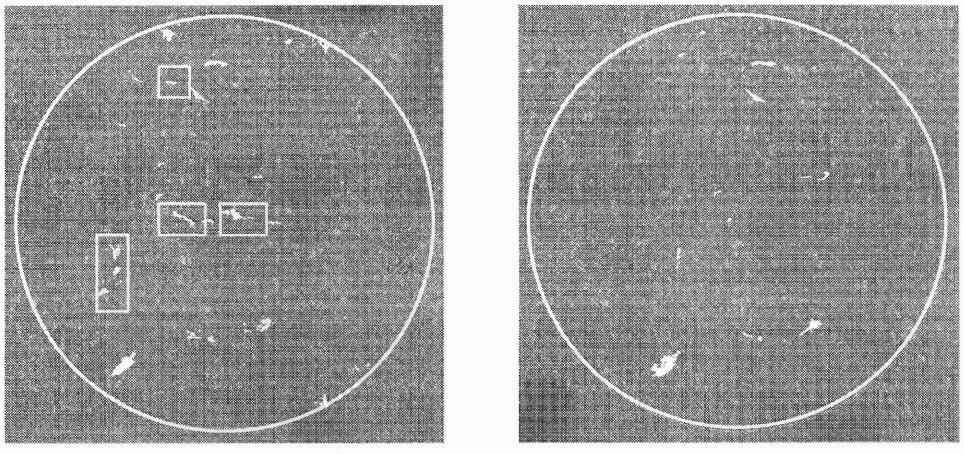
$$N = K \bullet S^{\alpha}$$



(a) 17:02 Feb. 22, 1999 Solar image and filaments detected



(b) 16:21 Feb. 23, 1999 solar image and filaments detected



(c) Filament disappearances between 17:02 Feb. 22 and 16:21 Feb. 23 1999

Figure 2.3 One example of filament disappearance detection.

where K value is a function of the "size threshold" selection. N and S follow a power law with a power index of $\alpha = -2.6$. Wang et al. [1998] analyze the relationship between the size distribution and the length of filaments that disappear. They have found that the distribution follows a law with a power index of –1.4. However, they use only the one-dimensional length distribution. It is reasonable to conclude that a two-dimensional distribution has a steeper power spectrum shape, because longer filaments appear to be proportionally wider than shorter filaments. As a comparison, the flux distribution of flare emissions in hard X-rays follows a power-law distribution, with a power index of –1.8 [Crosby, Ashwanden and Dennis, 1993].

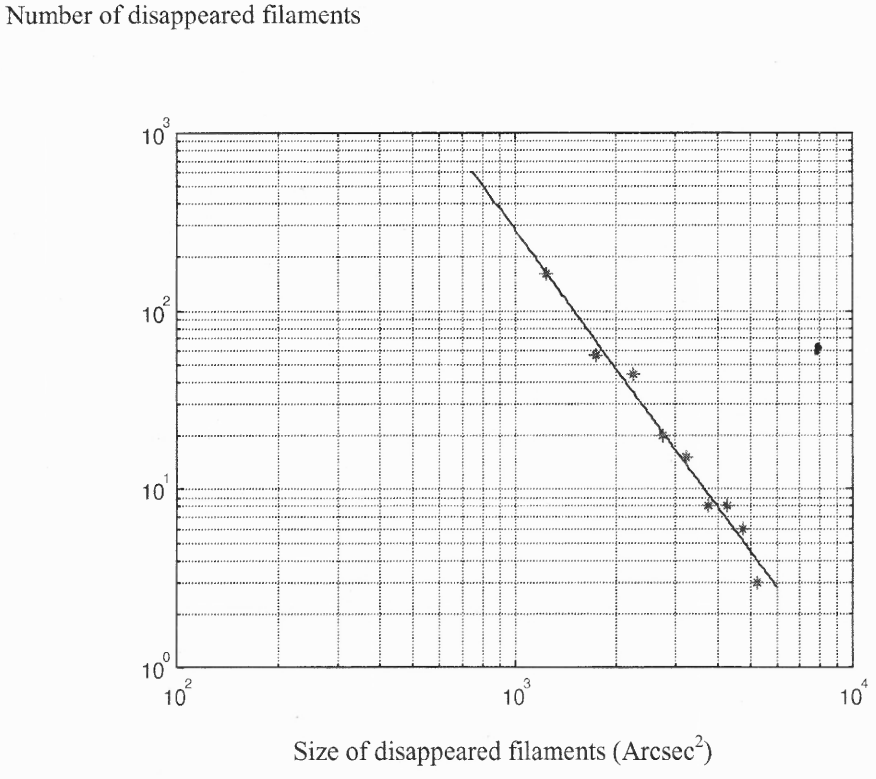


Figure 2.4 Size distribution of filaments disappeared.

2.4.3 Filament Frequency in Time Series

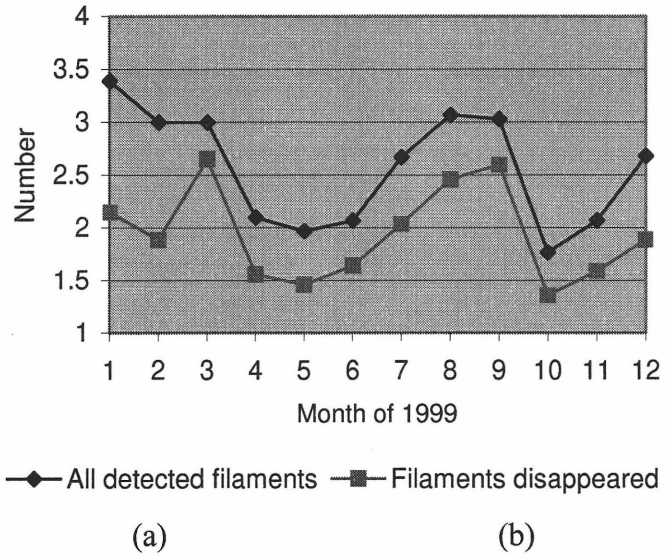


Figure 2.5 The number of filaments and filaments disappeared.

The number of all detected filaments as a function of time in 1999 is shown in Figure 2.5(a). The number of disappearing filaments as a function of time in 1999 is shown in Figure 2.5(b). The number of filaments peak in January/February and August/September. The time difference between these two peaks is about six months. In Figure 2.5(b), the number of filaments that disappeared, as a function of the time, is shown. More filaments disappeared in March, August and September and the time between the two most active months is five to six months. Bogart and Bai [1985] present a 152-day periodicity in the occurrence of solar flares. This work may have detected such a periodicity in the occurrence of filaments, although only two complete periods are shown.

2.4.4 Latitude Distribution of Filaments and Filaments Disappeared

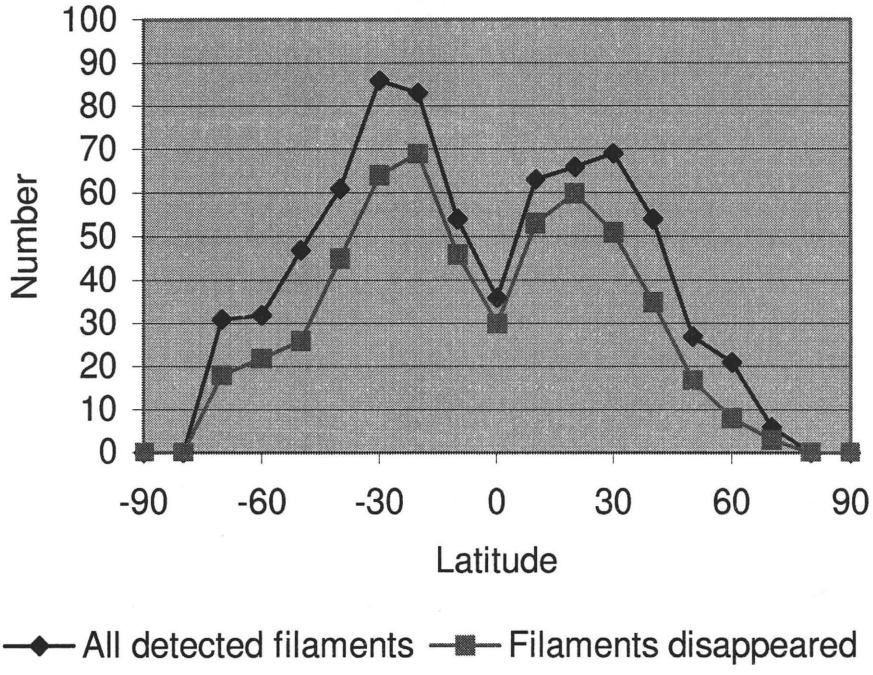


Figure 2.6 Distribution of filament positions.

This work can also analyze the latitude at which the filaments in the study occur. In Figure 2.6, the distribution of filament midpoint latitudes is plotted. It is noted that the location of filaments peaks at approximately ± 30 degrees. This corresponds very well with the Sun's active latitudes.

2.5 Summary

An efficient algorithm to automatically detect solar filaments is introduced in this work. This algorithm plays a key role in the detection of disappearing filaments at BBSO. "Intensity threshold" and region growing methods are combined in the algorithm.

"Intensity threshold" is used to detect the dark pixels that are filament candidates. A region growing method merges the detected pixels to form the filament areas.

Automatic filament detection is an important first step to real time filament disappearance detection. BBSO records one solar image every minute for a total of typically 600 per day. Due to the high running speed of the algorithm, it is possible to process at least one image every minute. A shell program has been developed to automatically process the images.

All full disk Hα images from 1999 have been analyzed to demonstrate the statistical results. The size distribution of filament disappearances follows a power law with a power index of –2.6. The time between the two most active periods of filament disappearances is about five to six months. The solar disk location of filaments peak at approximately ± 30 degrees.

Obviously, if a filament disappearance occurs near the solar disk center, it has the potential to generate a major geomagnetic event. The results are potentially useful in establishing a quantitative relationship between filament eruptions and geomagnetic activity. In addition, the latitude distribution and solar cycle variation may provide valuable information for the study of the solar dynamo. The future work is to process high resolution solar images from BBSO with this proposed algorithm. More accurate locations of filament events can be reported.

CHAPTER 3

THREE DIMENSIONAL SURFACE RECONSTRUCTION

3.1 Introduction

The problem of representing and reconstructing three-dimensional shapes has received an enormous amount of attention in computer vision and graphics research for the past decade. The interest arises at least in part because a useful shape theory would have applicability to a wide variety of fields such as design automation, manufacturing automation, terrain mapping, vehicle guidance, archeology, restoration of works of art, surveillance, and intelligent robots in general.

A vision system that makes use of an object model is referred to as a model-based vision system, and the general problem of identifying desired objects is referred to as object recognition. While there is no single definition of the object recognition problem, the objective is to identify a desired object in the scene and to determine its exact location and orientation.

An ideal vision system should be able to locate objects in a scene, assuming that: (1) objects may have arbitrary and complicated shapes or forms; (2) objects may be viewed from any direction, and (3) objects may be partially occluded by other objects. Specifically, such a system may be used in determining the location of grasp sites for a robot arm to manipulate an object, in the navigation of a robot or an autonomous vehicle, and in the assembly and inspection of parts in a manufacturing environment.

To design such systems, system designers must resolve the following issues: (1) the type of sensors for data collection, (2) the methods of constructing a necessary object model, (3) the means of describing the collected data and model, and (4) the methods of

matching the object descriptions obtained from the input data to that of the model [Arman and Aggarwal, 1993]. Sensors determine the resolution (the total number and frequency of sample points) and precision (the accuracy of each sampled point). More importantly, they determine whether the data provides 2D or 3D information of the scene. Models provide a priori knowledge of the vision system. Representations are used to describe the collected data and object model, a key issue in the field of computer vision. The representations dictate the matching strategy, robustness, and the system's efficiency. Also, the descriptions are used in the calculations of various properties of objects in the scene needed during the matching stage. Matching strategies are performed at run-time and must resolve many ambiguities that exist between data and model descriptions. Once the correct match has been determined, the orientation and translation of the located object, with respect to the model, can be calculated, completing the task of object recognition.

3.2 Issues of an Automatic 3D Object Model Construction System

Building models of unknown objects from sensory data has attracted the efforts of researchers. Four steps are necessary to build a surface description of an object using range data as shown in Figure 3.1.

Step 1: Data acquisition. The intensity camera is perhaps the most commonly used sensing module, measuring visible light. The output of the camera form a scene is digitized to provide a 2D array of numbers; each number corresponds to the average intensity sensed in a sampled, typically square area. Examples of other sensory modules include thermal cameras, which measure the emitted thermal radiation rather than the emitted visible light, and laser range scanners and sonar sensors, which are used to

calculate the distance to the objects in the scene. These sensors each provide information on a different aspect of their environment. The choice of the sensor is largely application dependent. For the task of 3D object recognition, various object dimensions and surface shape information are essential. Two general approaches exist to collect the necessary data. In the first approach, sensing modalities, such as intensity cameras, are used, and many depth cues are analyzed to recover the necessary 3D information. In the second approach, external energy sources, such as lasers, are projected onto the scene. While the first approach is preferred, the recovered data lacks the necessary resolution and precision for many common tasks. Therefore, the second approach is used most frequently in 3D object recognition.

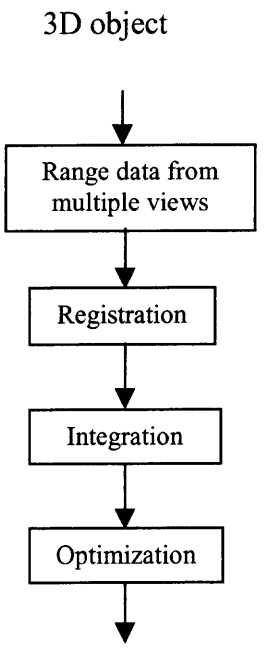


Figure 3.1 Overview of an automatic 3D object model construction system.

Several range views must be acquired from different viewpoints since a single view cannot capture all the surface information. Some preprocessing of the data may be required, such as error point deletion, low-pass filtering and removal of the data points that belong to other objects or background.

Step 2: Registration. The range views must be registered. The frame transformation between each pair of range views must be found. In general, each range view is in its own coordinate system. To model an object, a common reference frame in which all views can be transformed must be defined.

Step3: Integration. The range views should be integrated into a non-redundant model. For a model to be non-redundant, each element of surface in the scene must be described by a single geometric primitive. The single surface representation is often a polygon mesh.

Step 4: Optimization. The single surface representation can be better fit to the data. It may be further simplified, or the representation can be converted to some other format.

In the next four sections, the previous works done are reviewed in these four steps.

3.3 Data Acquisition

An important aspect of a computer vision system is its data acquisition module. This section introduces the schemes in which 3D information is acquired from the scene. The task is performed in one of two approaches: passive and active. In the passive approach, 3D information is inferred form the scene using existing energy in the environment, such as reflected light. In the active approach, the 3D information is derived by projecting external energy waves, such as sonar waves and laser light. 3D data recovered from

current passive approaches lacks the necessary precision and resolution for 3-D object recognition.

3.3.1 Passive Range Sensing

Passive-range sensing is also regarded as stereometry. Stereometry is a technique by which one can deduce a three-dimensional shape of an object from a stereoscopic image pair. The object of interest is an opaque surface in front of the camera. Depending on the reflectance of that surface, a portion of the light striking it is reflected, scattering in all directions. Some portion of the scattered light passes through the lens aperture and forms an image of the object at the image plane of the camera. The pixel cone's intersection with the object defines that region of the object to which the pixel corresponds. A portion of the light incident upon the pixel's image is scattered back into the lens aperture. All of this light is converged by the lens to fall upon the given pixel and thus to determine its gray level.

In addition to brightness, another value can be associated with the pixel in question. The distance from the center of the lens to the point p defines the range of this pixel. However if other surfaces lie behind the object, they are obscured. Thus the range of a pixel is the distance along its pixel cone form the center of the lens to the first opaque surface encountered. A range image is generated by assigning each pixel a gray level proportional, not to its brightness, but to the length of its pixel cone.

Figure 3.2 diagrams a dual camera configuration suitable for stereoscopic imaging [Castleman, 1996]. A three-dimensional coordinate system has its origin at the center of the lens of the left camera. In this figure, the optical axes of the two cameras are parallel

and lie in the XZ-plane. Under these conditions, the cameras are said to be boresighted. The Z-axis coincides with the optical axis of the left camera. Both camera lenses have focal length f, and they are separated by the distance d.

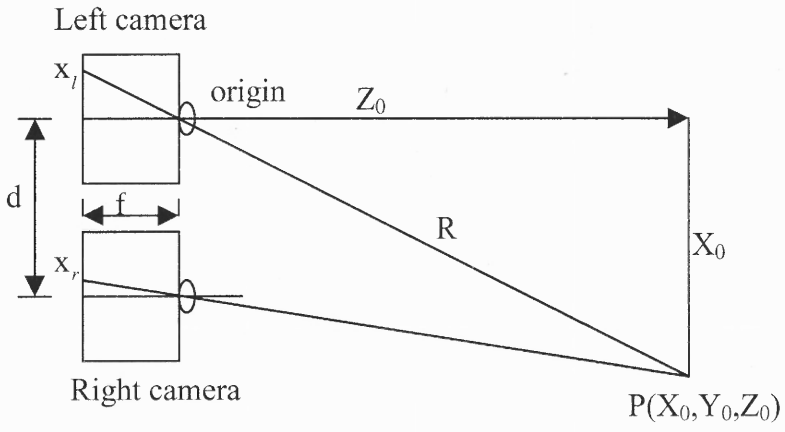


Figure 3.2 Stereoscopic imaging.

Suppose that the point P with coordinates (X_0, Y_0, Z_0) is situated in front of the cameras, casting an image on both image planes. Then, using similar triangles in the XZ-plane and YZ-plane, a line from P through the center of the left camera lens intersects the Z = -f (image) plane at

$$X_l = -X_0 \frac{f}{Z_0} \qquad Y_l = -Y_0 \frac{f}{Z_0} \tag{3.1}$$

Similarly, a line from P through the center of the right camera lens intersects the image plane at

$$X_r = -(X_0 + d)\frac{f}{Z_0} - d \qquad Y_r = -Y_0 \frac{f}{Z_0} \tag{3.2}$$

A two-dimensional coordinate system is now set up in each image plane. It is convenient to have each of them rotated 180 degrees from the main coordinate system, so as to counteract the rotation that is intrinsic to the imaging process. Thus

$$x_l = -X_l \qquad y_l = -Y_l \qquad x_r = -X_r - d \qquad y_r = -Y_r \tag{3.3}$$

Now the coordinates of the point images are

$$x_l = X_0 \frac{f}{Z_0} \qquad y_l = Y_0 \frac{f}{Z_0} \tag{3.4}$$

and

$$x_r = (X_0 + d)\frac{f}{Z_0} \qquad y_r = Y_0 \frac{f}{Z_0} \tag{3.5}$$

Rearrange equations (3.4) and (3.5)

$$X_0 = x_l \frac{Z_0}{f} = x_r \frac{Z_0}{f} - d \tag{3.6}$$

Solving this for Z_0 produces the normal-range equation.

$$Z_0 = \frac{fd}{x_r - x_l} \tag{3.7}$$

This equation relates the normal component Z_0 of range to the amount of pixel shift between the two images. Z_0 is a function of only the difference between x_r and x_l, and not their individual magnitudes. Since Z_0 must be positive, $X_r \geq X_l$. The numerator may be rather small compared to Z_0, implying that the denominator (the pixel shift) will be extremely small for large Z_0, thus small inaccuracies in determining the position of a feature in the two images can produce large errors in range calculations.

3.3.2 Active Range Sensing

Active-range sensing can be divided into two main classes. In the first class, the principle of triangulation is used. Each point in a scene is highlighted, using a sheet of light, and observed by the sensor. Then, using the known geometry of the imaging system, the distance of each highlighted point to the sensor is calculated. The second class of active-range sensors is known as time-of-flight sensors. A signal is emitted, and its return time is measured and used in calculating the distance [Arman and Aggarwal, 1993].

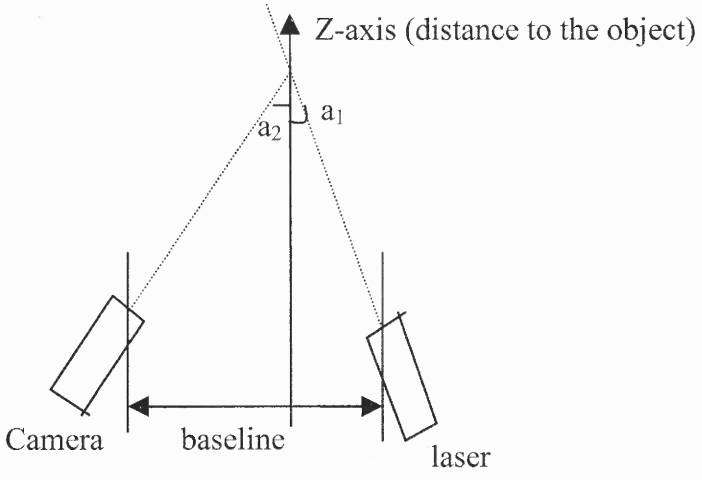


Figure 3.3 The geometry of a typical laser triangulation system.

Triangulation-based method uses a laser source that projects a sheet of light onto the scene, casting a line on the objects (Figure 3.3). A camera is positioned so that the laser line is visible. Using the known imaging geometry, i.e., the distance between the laser source and the camera (referred to as baseline), their angles (a_1 and a_2) with the z-axis (the axis along which depth is measured) and by applying the principles of triangulation, the distance of the illuminated points, along the cast laser line, to the baseline is calculated. Sweeping the sheet of light across a scene results in a range map.

The sweeping of the sheet is performed in one of two ways: a rotating reflector can be used to project the sheet of light, or the objects can be placed on a stage which slides by or rotates in front of the sheet of light. In all cases, the laser may not illuminate some areas, or the sensor may not be able to detect the illumination (i.e., the object concavities may occlude some areas of the object); and there are no data points at those locations of the scene. These areas of missing points are referred to as shadows. The shadow problem can be alleviated when using a second camera.

3.4 Range Image Registration

The second step in object surface reconstruction is image registration. Techniques to solve the registration problems fall into two categories. In the first one, the transformation relation between several views is known. The techniques rely on precisely calibrated data under these transformations. The second one develops techniques to estimate the transformation from data directly. In most cases, a coarse estimate of the transformation between each pair of range views is part of the available information.

The first type is often inadequate to construct a complete description of complex shaped objects because views are restricted to rotations or to some known viewpoints only. Therefore, people can not make use of the object surface geometry in the selection of observer viewpoints to obtain measurements. The second category is general and involves searching a huge parameter space. Hence, even with good heuristics, it may be computationally very expensive.

Chen and Medioni [1992] avoid the search in the view transformation space by assuming an initial approximate transformation for registration, which is improved with

an iterative algorithm that minimizes the distance from points in a view to a tangential places at corresponding points in other views. Suppose (p, q) is a pair of points from the two views representing the same surface point. It can be related to each other by one rigid spatial transformation, i.e., there exists a rigid transformation T, such that:

$$\forall \, p \in P, \exists \, q \in Q, \ \|Tp - q\| = 0 \tag{3.8}$$

where P and Q are two views of the same surface, Tp is the result of applying transformation T to p, and T is a transformation.

In Figure 3.4, Chen and Medioni [1992] approximate Q using its tangent plane S_j at q_j. The evaluation function can be written as :

$$e = \sum_{i=1}^{N} \left\| T\, p_i - q'_j \right\|^2 , \text{ with } \ q'_j = q / \min_{q \in S_j} \left\| T\, p_i - q \right\| \tag{3.9}$$

where S_j is the tangent plane of Q at q_j. q_j cannot be decided. They set the initial transformation T_0. Then they start an iterative process as an approximation. $q_j^{'k}$ is an approximation to q_j at each iteration. Since the distance from a point to a plane can be expressed as a linear function of the coordinates of the point, the iterative procedure can be formulated as follows:

$$e^k = \sum_{i=1}^{N} d_s^2 (T^k p_i, S_j^k) \tag{3.10}$$

with

$$S_j^k = \{ s \mid n_{qj}^k . (q_j^{'k} - s) = 0 \}, q_j^{'k} = (T^{k-1} l_i) \cap Q \tag{3.11}$$

where d_s is the signed distance from a point to a plane and n_{qj}^k is the surface normal vector of Q at $q_j^{'k}$. l_i is the line normal to P at p_i.

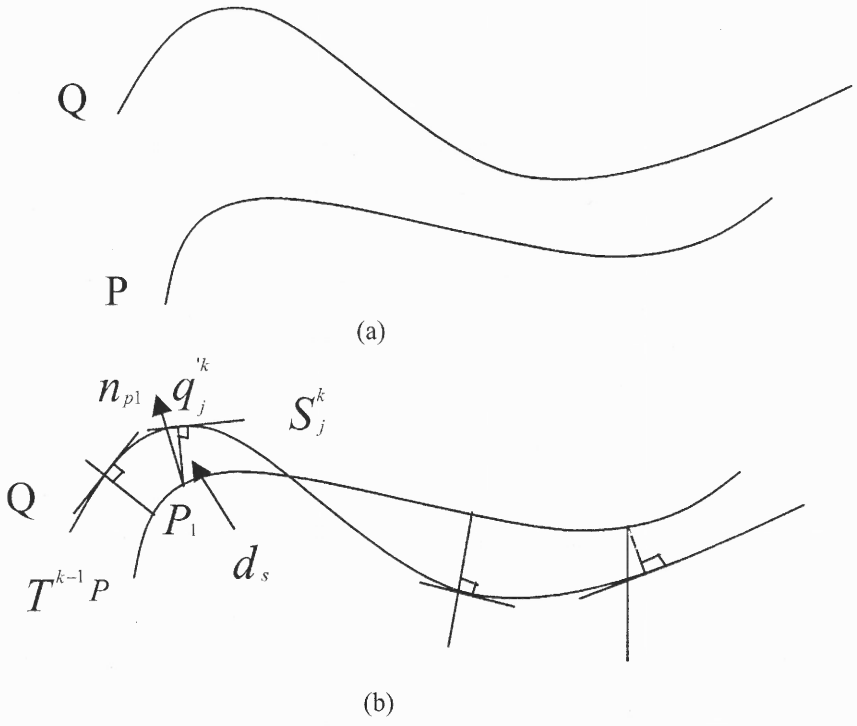


Figure 3.4 Distance measures between P and Q illustrated in 2D case [Chen and Medioni, 1992].
(a) Q and P before T^{k-1} is applied.
(b) Distance to the tangent plane of Q.

Besl and Mckay [1992] propose an iterative closest point (ICP) algorithm to handle the full six degrees of freedom for registration. The algorithm requires only a procedure to find the closest point on a geometric entity to a given point. It always converges monotonically to the nearest local minimum of a mean-square distance metric.

Bergevin et al. [1996] also make use of the iterated closest-point (ICP) algorithm. The algorithm considers the network of views as a whole and minimizes the registration errors of all views simultaneously. A set of N range views $\{V_1,V_2,...,V_N\}$ of a given scene is assumed. In order to integrate the surface information provided by the N range views, it is required to define the reference frame of one of these views as being the

world reference frame into which the integrated surface model is described. To each range view is associated a node in a network of views. An arc between two nodes stores the inter-frame transformation between the two associated views. All range views are interconnected by this network of inter-frame transformation; that is, for any pair of nodes, at least one path joins the two nodes. A path of inter-frame transformations in a network of views consists of a sequence of matrix multiplications. If the transformation matrices between the views are inexact, such a sequence of multiplications may result in an accumulation of registration errors. It is thus of primary importance to use a network topology that minimizes the lengths of the paths between the nodes. As shown in Figure 3.5, a well-balanced network of views is obtained. The registration errors in all transformation matrices are equally distributed.

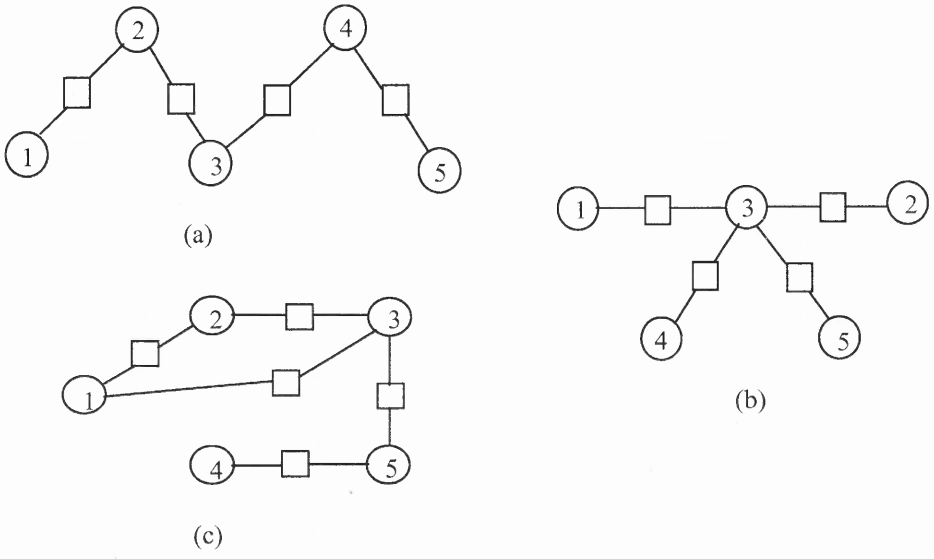


Figure 3.5 Three different types of network topologies. Each link joining two nodes represents a transformation matrix between two range views.

Blais and Levine [1995] define a function that measures the quality of the alignment between the partial surfaces containing in two range images as produced by a set of motion parameter. This function computes a sum of Euclidean distances between a set of

control points on one of the surfaces to corresponding points on the other. Let S_c be a set of control points taken form the total set of points in the first view. Let T be the transformation which takes a point in the first view and expresses it in the reference coordinate frame of the second. If $\vec{p}$ is a point in the first view, then T($\vec{p}$) is the same point expressed in the second view's coordinate frame. $\vec{d}$ is a translation matrix and R is a rotation matrix. T($\vec{p}$) is simply given by:

$$T(\vec{p}) = R\ \vec{p} + \vec{d} \tag{3.12}$$

Let C(·) be the correspondence function defined by the camera's inverse calibration equations. Given a transformation T, the cost function for T is as follows,

$$\text{cost (T)} = \sum_{p \in S_c} d(T(\vec{p}), C(T(\vec{p}))) \tag{3.13}$$

where d(·) is the 3D Euclidean distance (L_2 norm) between two points. The cost function is an indication of the registration quality of transformation T.

Many literature articles report the registration methods according to ICP algorithm. The heart of the ICP approach is in finding a rigid transformation that minimizes the least-squared distance between the point pairs. Horn [1987] describes a closed-form solution to this problem that is linear in time with respect to the number of point pairs. Horn's method finds the translation vector T and rotation R such that:

$$E = \sum_{i=L}^{n} \left| A_i - R(B_i - B_c) - T \right|^2 \tag{3.14}$$

is minimized where A_i and B_i are given pairs of positions in 3-D space and B_c is the centroid of B_i. Horn shows that T is just the difference between the centroid of the points A_i and the centroid of the points B_i. R is found by constructing a cross-covariance matrix

between centroid-adjusted pairs of points. The final rotation is given by a unit quaternion that is the eigenvector corresponding to the largest eigenvalue of a matrix constructed form the elements of this cross-covariance matrix [Horn, 1987; Besl and Mckay, 1992; Turk and Levoy, 1994].

Although it is theoretically easy to estimate the motion parameters between two sets of 3D points, practical considerations complicate the implementation of the system. In stereo imagery, the range values estimated are subject to a great deal of uncertainty due primarily to quantization of disparity [Aggarwal and Nandhakumar, 1988]. More robust formulations of the problem of motion estimation using sequences of stereo images have been proposed in [Huang and Blostein, 1985; Lin et al., 1986].

3.5 Range Image Integration

A set of registered range images should be integrated into a non-redundant surface model. Surface reconstruction from the range images has been an active area of research for several decades. There are some good solutions available in literature. However, until now there is no general method that handles any special surface reconstruction problems. Bolle and Vemuri [1991] review the 3-D surface reconstruction methods. In general, computer representations of 3-D object shape fall into two categories: surface representation and volume representation. Surface representation is concerned only. The strategies have proceeded along two basic directions: reconstruction from unorganized points, and reconstruction that exploits the underlying structure of the acquired data [Curless and Levoy, 1996]. Here some new and efficient solutions in the integration are introduced.

One of two different philosophies to perform surface-based integration of a set of multiple registered range images is to directly compute surface models on sets of unorganized 3D points. A major advantage of the unorganized point algorithms is the fact that they do not make any prior assumptions about connectivity of points. In the absence of range images or contours to provide connectivity cues, these algorithms are the only recourse.

The goal of surface reconstruction in [Hoppe et al., 1992] is to determine a surface M' approximating an unknown surface M by using a sample X and information about the sampling process, for example, bounds on the noise magnitude δ and the sampling density ρ. Their surface reconstruction algorithm consists of two stages. In the first stage they define a function f: $D \rightarrow IR$, where $D \subset IR^3$ is a region near the data, such that f estimates the signed geometric distance to the unknown surface M. The zero set Z(f) is the estimate for M. In the second stage they use a contouring algorithm to approximate Z(f) by a simplicial surface. The key ingredient to defining the signed distance function is to associate an oriented plane with each of the data points. These tangent planes serve as local linear approximations to the surface. Although the construction of the tangent planes is relatively simple, the selection of orientations so as to define a globally consistent orientation for the surface is one of the major obstacles facing the algorithm. The tangent planes are used to define the sighed distance function to the surface. Hoppe's algorithm has four steps.

The first step is to compute an oriented tangent plane for each data point. The tangent plane associated with the data point is represented as a center point together with a unit normal vector. The signed distance of an arbitrary point to a tangent plane can be

decided. The center and normal of the tangent plane are determined by gathering together the k points of X nearest to the data point. The second step is to get the consistent tangent plane orientation. Suppose that two data points x_1 and $x_2 \in X$ are geometrically close. When the data is dense and the surface is smooth, the corresponding tangent planes are nearly parallel. If the planes are consistently oriented, then the product of two unit normal equals to +1; otherwise, one of two unit normals should be flipped. The difficulty in finding a consistent global orientation is that this condition should hold between all pairs of sufficiently close data points. The third step is to define the signed distance function. The signed distance from an arbitrary point p to a known surface M is the distance between p and the closest point on M, multiplied by ±1, depending on which side of the surface p lies. In reality M is not known, but oriented tangent planes can mimic this procedure. First, the tangent plane is found whose center o is the closest to p. This tangent plane is a local linear approximation to M. Thus, the signed distance f(p) to M is the signed distance between p and its projection onto the tangent plane, or,

$$f(p) = \text{dist}(p) = (p - o) \cdot n \tag{3.16}$$

where n is the unit normal of point p. The fourth step is contour tracing. It is the extraction of an isosurface from a scalar function. Hoppe et al. implement a variation of a marching cubes algorithm that samples the function at the vertices of a cubical lattice and finds the contour intersections within tetrahedral decompositions of the cubical cells.

The algorithm reconstructs the surface with or without boundary from data points scattered on or near the surface. It is capable of automatically inferring the topological type of the surface, including the presence of boundary curves. Hoppe et al. also present a new class of piecewise smooth surface representations based on subdivision [Hoppe et

al., 1994]. The main purpose of moving from piecewise linear to piecewise smooth surfaces is to obtain more accurate and concise models of unorganized point sets.

The other philosophy of performing surface-based integration assumes that a parametric surface description can be derived from a single range image. An integrated surface model can then be obtained by merging multiple surface descriptions using an appropriate technique. Averaging the measurements of several range images yields a more accurate integrated model when the registration error can be neglected with respect to the acquisition error. Few reports describe the techniques to merge multiple surface descriptions derived from range images. Soucy and Laurendeau [1995] present a solution to the problem of range view integration. They assume that reliable view transformations are available and proceed by building a set of triangulations with the subsets of the common surface segments between all pairs of views, and connecting them to output a global triangulation. They propose an approach for measuring the level of redundancy in a set of range images through the use of the Venn diagram. Spatial Neighborhood Test (SNT) and Surface Visibility Test (SVT) are presented in computing the common surface segments between two range views.

Dorai et al. [1998] describe a weighted averaging algorithm instead of the simple averaging algorithm. Each pixel in the input image is associated with a weight that is essentially the distance of the pixel to the nearest boundary. However, with the simple averaging algorithm, the depth value of each pixel in the merged image is calculated by averaging the depth values of the corresponding pixels in two images. Figure 3.6 specifies two different integration algorithms.

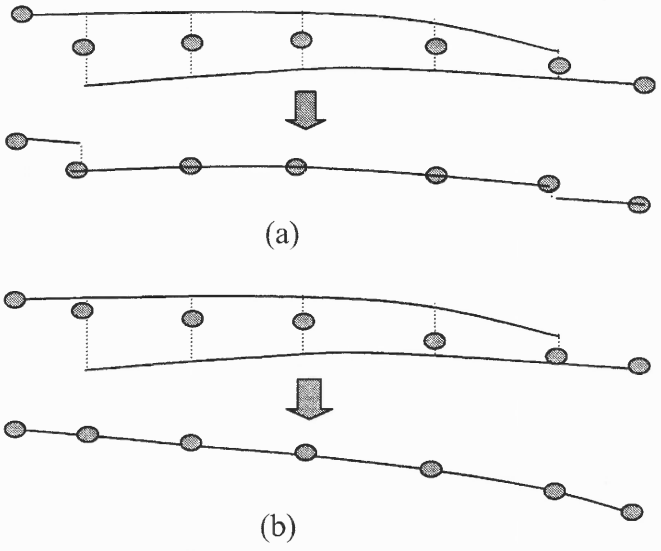


Figure 3.6 Comparison of integration algorithms [Dorai et al., 1998].
(a) Integration using simple averaging.
(b) Integration using weighted averaging.

Turk and Levoy [1994] present one efficient integration method. After aligning the meshes with each other using a modified ICP algorithm, they zipper together adjacent meshes to form a continuous surface that correctly captures the topology of the object and then compute local weighted averages of surface positions on all meshes to form a consensus surface geometry. Zippering two triangle meshes consists of three steps:

(1) Remove overlapping portions of the meshes;

(2) Clip one mesh against another; and

(3) Remove the small triangles introduced during clipping.

Figure 3.7 specifies the mesh clipping introduced by [Turk and Levoy, 1994]. Cappellini et al. [1991], Bolle et al. [1991] and Goshtasby [1998] review the model construction methods.

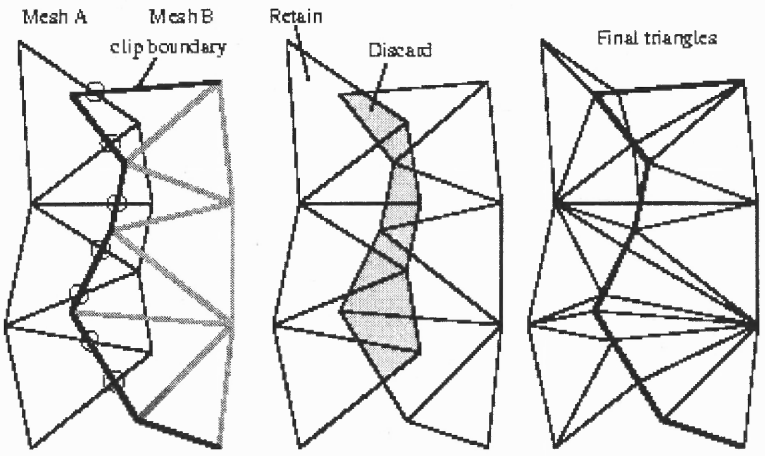


Figure 3.7 Mesh A is clipped against the boundary of mesh B. Circles (left) Show intersection between edges of A and B's boundary. Portions of triangles from A are discarded (middle) and then both meshes incorporate the points of intersection (right) [Turk and Levoy, 1994].

3.6 Surface Simplification

Surface simplification becomes increasingly important as it becomes possible to create models of greater and greater detail thereby requiring more and more memory space. Many techniques can generate surface models consisting of millions of polygons.

Simplification is useful in order to make storage, transmission, computation, and display more efficient. A compact approximation of a shape can reduce disk and memory requirements and can speed network transmission. It can also accelerate a number of computations involving shape information, such as finite element analysis, collision detection, visibility testing, shape recognition, and display. Reducing the number of polygons in a model can make a difference between slow and real time display.

Heckbert and Garland [1997] survey previous work on surface simplification in depth. Their survey covers many methods before 1997 and some in 1997. Cignoni et al. [1998] present a survey and a characterization of the fundamental mesh simplification

algorithms. Erikson [1996] also provides a foundation and overview of this research field and discusses the details of how existing algorithms attempt to solve the mesh simplification problem.

Here polygonal surfaces are taken as input and produce polygonal surfaces as output. The polygons in this work are typically planar triangles. The surface simplification algorithm can be categorized into two classes: manifold surfaces and non-manifold surfaces.

In manifold surfaces, every edge is only shared by two triangles. A two-stage method for multi-resolution wavelet modeling of arbitrary triangulated polyhedra was developed by [Eck et al., 1995]. The method can be applied to any triangulated manifold with boundary. The approach first constructs a base mesh that is a triangulated polyhedron with the same topology as the input surface. It then uses repeated quaternary subdivision of the base mesh to construct a new mesh that approximates the input surface very closely. A multi-resolution model of the new mesh is then built using wavelet techniques, after which an approximation at any desired error tolerance can be quickly generated.

Schroeder et al. [1992] develop a general vertex decimation algorithm primarily for use in scientific visualization. Their method takes a triangulated surface as input, typically a manifold with boundary. The algorithm makes multiple passes over the data until the desired error is achieved. On each pass, all vertices that are not on a boundary or crease that have error below the threshold are deleted, and their surrounding polygons are retriangulated (Figure 3.8). The error at a vertex is the distance from the point to the approximation plane of the surrounding vertices.

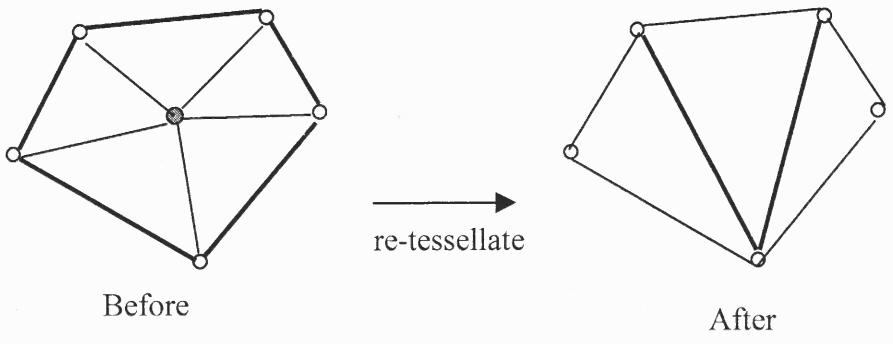


Figure 3.8 Vertex decimation. The target vertex and its adjacent triangles are removed. The resulting hole is then re-tessellated.

Soucy and Laurendeau [1996] also develop a vertex decimation algorithm. Their application is the construction of surface models from multiple range views. On each pass, the vertex with the least error is deleted, and its neighborhood (the set of adjacent triangles) is re-triangulated. The process stops when the error rises above a specified tolerance or the desired size of a model is achieved.

Turk [1992] takes a triangulated surface as input, sprinkles a user-specified number of points on these triangles at random, and uses an iterative repulsion procedure to spread the points out nearly uniformly. The points remain on the surface as they move around. After these points are inserted into the original surface triangulation, the original vertices are deleted one by one, yielding a triangulation of the new vertices that has the same topology as the original surface. Turk also demonstrated an improved variant of this technique that groups points most densely where the surface is highly curved.

Hoppe et al. [1993] develop an optimization-based algorithm for general 3-D surface simplification. Their method takes a set of points and an initial, fine triangulated surface approximation to those points as input, and outputs a coarser triangulation of the

points with the same topology as the input mesh. The method attempts to minimize a global energy measure consisting of three terms: a complexity term that penalizes meshes with many vertices, an error term that penalizes geometric distance of the surface from the input points, and a spring term that penalizes long edges in the triangulation. The method proceeds in three nested loops, the outermost one decreasing the spring constant, the middle one doing an optimization over mesh topologies, and the inner one doing an optimization over geometries. The topological optimization uses heuristics and random selection to pick an edge and either collapse, split, or swap it. The geometric optimization uses nonlinear optimization techniques to find the vertex positions that minimize the global error for a given topology. Topological changes are kept if they reduce the global error; otherwise discarded. In other words, the method makes repeated semi-random changes to the mesh, keeping those that allow better fit and/or a simpler mesh. Hoppe [1996] develops the construction of progressive meshes. It is a simplified version of the above algorithm. Rather than performing a more general search, it simply selects a sequence of edge contractions. The algorithm uses essentially the same error formulation of the earlier method, although it is augmented to handle surface attributes such as colors.

The simplification technique of [Cohen et al., 1996] generates a hierarchy of level-of-detail approximations for a given polygonal model. Their approach guarantees that all points of an approximation are within a user-specifiable distance ε from the original model and that all points of the original model are within a distance ε from the approximation. The algorithm can be used if it is critical that the approximate model lies within some distance of the original model and that its topology remains unchanged.

Ciampalini et al. [1997] propose multiresolution decimation method based on global error. JADE, the simplification solution has been designed to provide both increased approximation precision based on global error management, and multiresolution output. With a small increase in memory, which is needed to store the multiresolution data represention, they can extract any level of detail representation from the simplification results in an efficient way.

Conventionally, in order to produce high-quality simplified polygonal models, one must retain and use information about the original model during the simplification process. Lindstrom and Turk [1998] demonstrate a simplification method without the need to compare against information from the original geometry while performing local changes to the model. They use edge collapses to perform simplification, as do a number of other methods. They select the position of a new vertex so that the original volume of the model is maintained; and then minimize the per-triangle change in volume of the tetrahedra swept out by those triangles that are moved. They also maintain the surface area near boundaries and minimize the per-triangle area changes. Calculating the edge collapse priorities and the positions of new vertices requires only the face connectivity and vertex locations in an intermediate model. This approach is memory efficient, allowing the simplification of very large polygonal models.

Algorri and Schmitt [1996] develop an algorithm for simplifying closed, dense triangulations resulting from surface reconstruction. Their algorithm begins with a pre-processing phase which smoothes the initial mesh by swapping edges. After this initial smoothing, every edge whose dihedral angle exceeds some user-specified planarity threshold is classified as a feature edge. Each vertex is subsequently labeled according to

its number of incident feature edges. An independent set of edges connecting "0" vertices is collected, and all the edges are collapsed simultaneously. This simplification phase is followed by a smoothing phase. If further simplification is desired, edges are reclassified and the process outlined above is repeated. Since only edges in mostly planar regions are selected for decimation, the basic step does not simplify "characteristic curves" and there is always a single vertex left in the midst of planar regions.

The above algorithms handle the manifold surface simplification. The most general class of surfaces is the non-manifold surface, which permits three or more triangles to share an edge and arbitrary polygon intersections. Relatively few surface simplification algorithms can handle models of this generality. Garland and Heckbert [1997] develop an algorithm for simplifying surfaces based on iterative vertex-pair contractions. A pair contraction is a natural generalization of edge contraction (Figure 3.9) where the vertex pair need not be connected by an edge (Figure 3.10). A 4x4 symmetric matrix Q_i is associated with each vertex V_i. The error at the vertex is defined to be V^TQV, and when a pair is contracted, their matrices are added together to form a matrix for the resulting vertex.

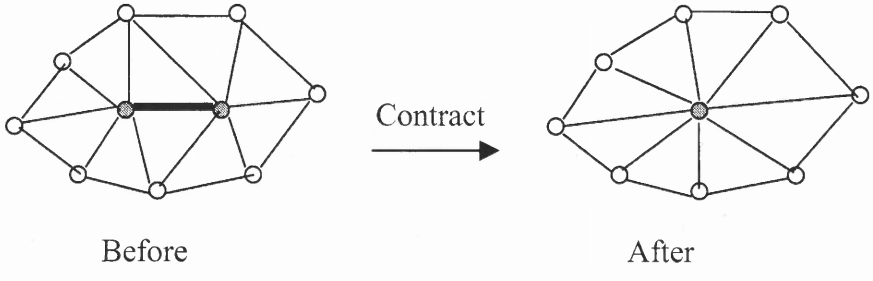


Figure 3.9 A simple edge contraction. The highlighted edge is contracted into a single point. Two triangles become degenerated and are removed during the contraction.

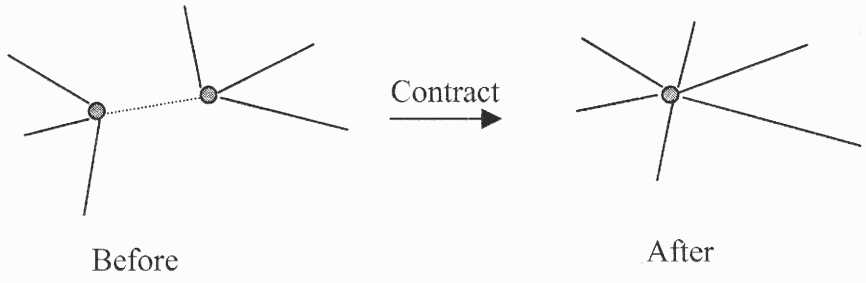


Figure 3.10 Pair contraction joining unconnected vertices. The dashed line indicates the two vertices being contracted together.

Low and Tan [1997] present a practical technique to automatically compute approximations of polygonal representation of 3D objects. It is based on a vertex-clustering method. The approach groups vertices into clusters and, for each cluster, computes a new representative vertex. The advantages of a vertex-clustering technique are its low computational cost and high data reduction rate, and thus suitable for use in interactive applications. But neither topology nor shape is preserved well.

There is related work in fitting curved surfaces to a set of points on a surface. Curved surface primitive and the subdivision surface are fit to points in 3-D with very nice results [Hoppe et al., 1994]. They introduce a new class of piecewise smooth surface representations based on subdivision.

CHAPTER 4

MESH SIMPLIFICATION WITH AVERAGE PLANES FOR 3D IMAGES

As discussed in Chapter 3, mesh simplification is a very important stage in three dimensional image reconstruction. In most cases, a model consists of a large number of triangles. It is difficult to render, transmit and store such model data. In this work, a mesh simplification method is proposed based on computing the average planes combined by adjacent vertices of every edge and two edge vertices. RMS (root mean square) distance error of all these vertices of the edge to the average plane is computed. Two RMS error values, the edge length and one penalty item are combined to produce the final error metric. According to this value, the curvature of the facet can be decided. The edge with the least error metric is selected to collapse. One open mesh and two closed meshes are used to test the efficiency of the proposed method. The algorithm is compared with other major mesh simplification algorithms using Metro tool [Cignoni et al., 1998].

4.1 The Curvature Metric

This work applies edge collapse method to mesh simplification. The edge in the least curvature facet is selected first and collapsed. How to decide the curvature is a key factor in this algorithm. Two vertices of every edge have adjacent vertices (surrounding vertices). In the method of [Schroeder et al., 1992], one approximation plane can be computed from the surrounding vertices. This method is used to obtain the average plane as described below.

An average plane is constructed by the triangle normal, n_i, center, x_i, and area, A_i,

$$N = \frac{\sum n_i A_i}{\sum A_i} \qquad n = \frac{N}{|N|} \qquad x = \frac{\sum x_i A_i}{\sum A_i} \tag{4.1}$$

where the summation is over all adjacent triangles.

In Figure 4.1, an average plane is constructed by the surrounding vertices and edge vertices.

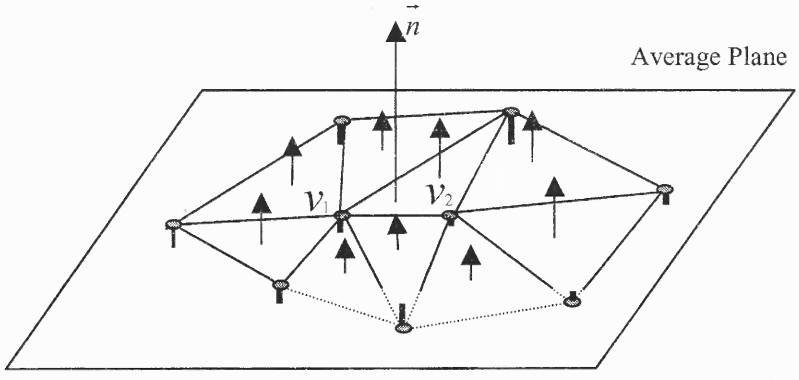


Figure 4.1 The average plane and vertices.

After an average plane is obtained, RMS (root mean square) is applied as the distance error measure. In the previous mesh simplification method [Schroeder et al., 1992] only the distance from candidate vertex to the average plane is considered. This may cause large errors. In Figure 4.2, although d_I and d_{II} are small, actually the curvature around the edge is large. The edge should not be collapsed first.

The above observation motivates us to set two curvature metrics E_1 and E_2. E_1 is the RMS distance error value between all surrounding vertices and the average plane, and E_2 is the RMS distance value between two edge vertices and the average plane.

$$E_1 = \frac{1}{n}\sqrt{\sum_{i=1}^{n} d_i^2} \tag{4.2}$$

$$E_2 = \frac{1}{2}\sqrt{d_I^2 + d_{II}^2} \tag{4.3}$$

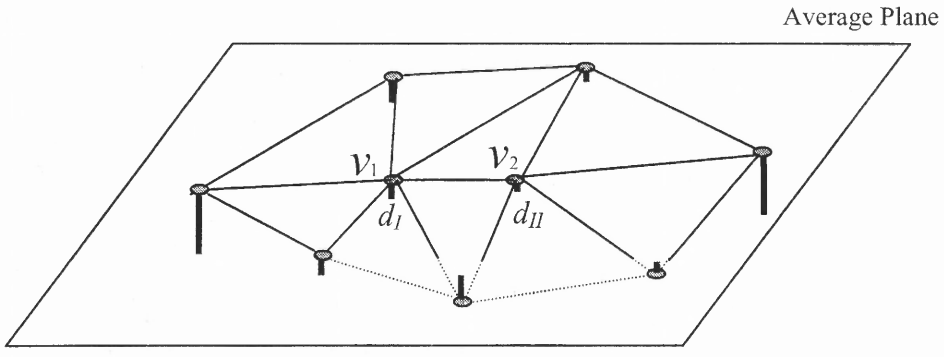


Figure 4.2 The edge on the larger curvature surface.

Consider the following cases:

(1) E_1 is large and E_2 is small;

(2) E_1 is small and E_2 is large;

(3) E_1 and E_2 are both large; and

(4) E_1 and E_2 are both small.

In cases 1-2, the facet has big curvature value and the edge should be kept. In case 3, the facet has bigger curvature value than those in cases 1 and 2. Thus the edge must be kept to the last order. In case 4, the curvature value is small. The edge can be collapsed to one vertex.

Considering all the situation of the error metric, E_1 and E_2 are combined to get the error metric in the algorithm. Suppose it is E:

$$E = E_1 + \alpha \cdot E_2 \tag{4.4}$$

where α is a weight used to signify the relative importance of E_1 and E_2 in the total error metric. Edge vertices always lie around the center of an average plane.

4.2 The Algorithm

The algorithm starts with an original triangulation model. The error metric of every edge is computed and ordered from the least to the largest. According to the error metric, the model is successively simplified. A candidate edge is collapsed to a vertex. The surrounding vertices are connected with this vertex to form new triangles. The error metrics of these new edges are computed.

4.2.1 The Edge Length

The length of an edge needs to be included in the error metric. If models have the same planar surface, those with the shorter edges should be collapsed. The error metric is thus changed to:

$$E = \text{length of edge} \cdot (E_1 + \alpha \cdot E_2) \tag{4.5}$$

Under this error metric, one shorter edge on less planar surface may be collapsed earlier than one longer edge on a planar surface. Because longer edges affect the model geometry more than shorter ones do, the model geometry can be kept better by considering the length of an edge in an error metric.

4.2.2 The Boundary in Open Mesh

Each vertex may be assigned one of five possible classifications: simple, complex, boundary, interior edges and corner edges [Schroeder et al., 1992]. A simple vertex is surrounded by a complete cycle of triangles, and each edge that uses the vertex is shared by exactly two triangles. Otherwise if some edge is not used by two triangles or if the vertex is used by a triangle not in the cycle of triangles, then the vertex is complex.

A mesh with complex vertices belongs to a non-manifold case and is thus not considered in this work. A vertex that is on the boundary of a mesh, i.e., within a semi-cycle of triangles, is a boundary vertex.

This work applies an edge collapse method instead of vertex collapse. It sets up one condition to decide the boundary edge. In manifold cases, every edge at most shares two vertices. More than two triangles sharing the edge in a mesh lead to a non-manifold case. As shown in Figure 4.3, no boundary edge is shared by more than one triangle.

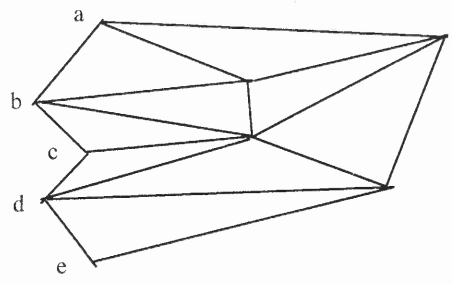


Figure 4.3 The boundary edges *ab, bc, cd* and *de*.

Vertices *a* and *b* of boundary edge *ab* shared by two triangles *abc* and *abd* in Figure 4.4 lead to a non-manifold case.

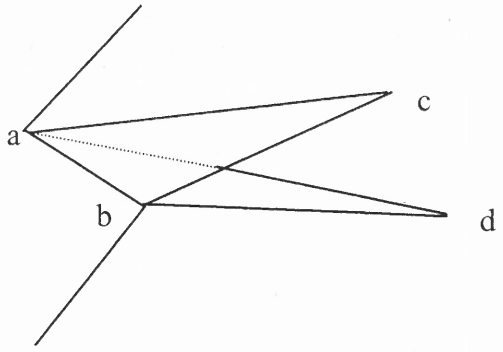


Figure 4.4 Boundary edge *ab* is shared by two triangles *abc* and *abd*.

The proposed algorithm starts with the whole model. Every edge is checked by the condition. If it is a boundary edge, two vertices are specified. In a simplification process, if an edge is combined with two boundary vertices, the error metric is changed to a larger number by adding one penalty number and changing the weight α to a large value, e.g., 100. If an edge is combined with one boundary vertex and one non-boundary vertex, weight α of its error metric should also be changed to 100.0. When an edge collapses to one vertex, its boundary vertices must be kept as the vertex placement.

Considering boundary edges, the error metric becomes:

$$E = \text{length of edge} \cdot (E_1+\alpha \cdot E_2) + \text{Penalty} \tag{4.6}$$

where penalty value is set to 1000.0 if both two edge vertices are boundary vertices, 0 otherwise. The weight α is set to 100.0 if any one of the edge vertices is boundary vertex, α=2.0 otherwise.

4.2.3 The Vertex Placement

When an edge is selected to collapse, some ways are needed to find a new vertex position. There are two different vertex placement policies, i.e., subset placement and optimal placement. There is tradeoff between space efficiency and model approximation quality.

Subset placement is a simple way. One vertex of the edge is chosen as the new vertex. After the average plane is computed, the vertex with smaller distance to the average plane is selected. Another vertex is contracted to this one. Under this method, all new vertices are selected from the subset of the original vertices.

In the optimal placement, the optimal position can be found after taking partial derivatives based on an error metric. However, a unique optimal position may not exist. A subset placement strategy is adopted. It is simple and can save significantly in storage. By adopting the convention that contractions are always ordered such that the result of contracting the pair (V_i, V_j) is to move V_j to the position of V_i, no additional storage is required to specify the new vertex position. As specified before, the boundary vertices are set to be the new vertex placement.

4.2.4 Mesh Consistency Check

An error metric is used to select the candidate edge to collapse and the vertex placement determines the new location of a resulting vertex. However, it is possible that one selected collapse may introduce mesh inconsistency. In order to solve this problem, some consistency checks can be set up in an edge collapse process. If the collapse fails one of the check, one large penalty item can be added to its error metric.

One of the most important checks is the triangle interior angle limitation check. In some application, sliver triangles, with one very small interior angle, are undesirable. Gueziec [1996] suggests a measure of triangle compactness

$$\gamma = \frac{4\sqrt{3}\omega}{l_1^2 + l_2^2 + l_3^2} \tag{4.7}$$

where l_i is the length of the ith edge of a triangle and ω is its area. For an equilateral triangle γ=1, and γ approaches 0 for a triangle whose vertices are co-linear. Several thresholds can be set. If the compactness of the adjacent triangles of two edge vertices

and the resulting triangles after the edge collapse fall below a certain threshold, a suitable penalty number is added to the edge error metric.

Another consistency check is the neighbor vertex number limitation of every vertex. Using the average plane and RMS distance error metric, it is possible that many edge error metrics are equal to 0. If a mesh has a large flat surface, for example the fandisk mesh in the following discussion, additional check has to be made besides the sliver triangle check. Otherwise a vertex may connect too many vertices in the flat surface and bring out mesh inconsistency problem such as mesh inversion. In Figure 4.5, one undesirable new edge *ef* causes mesh topology change.

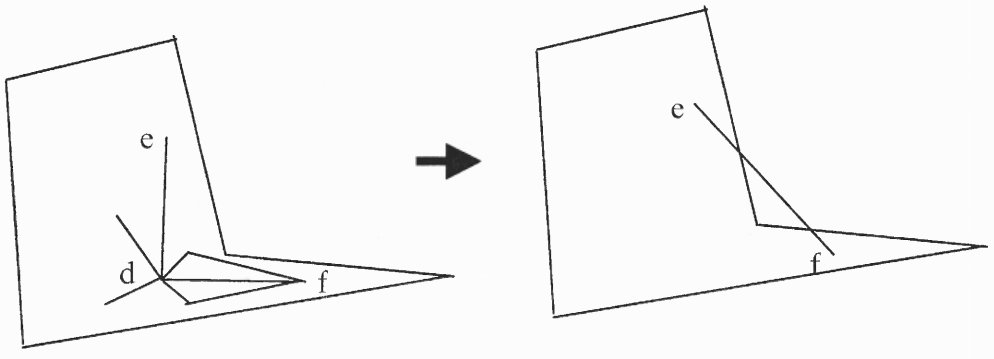


Figure 4.5 Collapse of an edge *df* that causes mesh inconsistency.

Vertex connectivity number limitation can be used to solve this problem. If connectivity number of either of two edge vertices is larger than 10 (more than 10 edges radiating from the vertex) or the summation of the connectivity numbers of two edge vertices is larger than 16, one large penalty item is added to this edge error metric. This check can also prevent the sliver triangles.

4.2.5 Basic Data Structure

The proposed algorithm is implemented in C++ and OpenGL. The basic data structures for the meshes and simplification process are briefly described. The triangle and vertex classes are specified as follows.

```
class Triangle {
 public:
        Vertex      * vertex[3];
        Vector      normal;          // unit normal
        Vector      center;  // the triangle center
        double      area;    // the triangle area
        Triangle(Vertex *v0,Vertex *v1,Vertex *v2);
        ~Triangle();
        void        ComputeNormal();
        void        ComputeArea();
        void        ComputeTriangleCenter();
    };

class Vertex {
 public:
    Vector    position;
    float     collapse_cost ;  //  the minimum cost to
                                    collapse the edge associated with the vertex
    int       num;     // place of vertex
    int    boundary; // to specify whether it is
                        boundary vertex
    List<Vertex *>  neighbor; // adjacent vertices
    List<Triangle *> face;    // adjacent triangles
    Vertex(Vector v,int _num);
     ~Vertex();
    };
```

The model is represented by an adjacency graph structure. The vertices, edges and faces are all explicitly represented and linked together. In *Vertex* class, the adjacent vertices and triangles are specified. *Collapse_cost* is the minimum cost to collapse one edge among all the edges associated with the vertex. *Triangle* class mainly includes the

functions to compute the triangle center, normal and area. These values are used to obtain an average plane. The algorithm can be summarized as follows:

(1) Input the 3-D data set;

(2) Compute the collapse cost of each edge;

- Select all surrounding vertices of the edge;
- Compute the average plane;
- Compute the distance from the vertices to the average plane;
- Compute E_1, E_2 and E.

(3) Place all the edges in a heap with the minimum cost edge on the top;

(4) Iteratively remove the pair (V_1, V_2) with the least cost from the heap, collapse the edge, update the adjacent triangles of removed vertex, and compute the E value of these new edges.

The mesh collapses to 0 vertex. All collapse processes are stored. Similar to Hoppe's Progressive Mesh method, the program needs to run only once. Different vertex number meshes are available.

4.3 The Results

Three sets of data, Stanford Bunny, Fandisk and ChristmasMan are used to test the developed algorithm. The first two are closed mesh while the third one is open mesh. In Figure 4.6, the original Bunny model has 69,451 triangles (upper left). The approximation models have 17,304, 6,894, 1,360 triangles respectively. The running time is about 376 seconds to finish the entire simplification process. The algorithm is also tested using fandisk with sharp edges and the ChristmasMan model with open mesh. The

results are shown in Figures 4.7 and 4.8. ChristmasMan model has a large number of faces. The simplification process takes about 37 minutes.

4.4 Simplification Algorithm Comparison

A uniform and general tool is adopted for the evaluation of approximation accuracy of this proposed algorithm and other simplification algorithms. Cignoni et al. [1998] develop Metro tool, which can measure the actual geometric "difference" between the original and simplified meshes.

Metro numerically compares two triangle meshes S_1 and S_2, which describe the same surface at different levels of detail. It requires no knowledge of the simplification approach adopted to build the reduced mesh. Metro evaluates the difference between the two meshes on the basis of the approximation error. It adopts an approximate approach based on surface sampling and the computation of point-to-surface distances. Metro returns both numerical and visual evaluation of surface meshes "likeness". The numerical results include mesh surface area, feature edges total length, mean and maximum distances between meshes and mesh volume. Error is also visualized by rendering the higher resolution mesh with a color for each vertex that is proportional to the error [Cignoni et al., 1998].

4.4.1 Terminology

Cignoni et al. [1998] define some terms used in Metro tool. The approximation error between two meshes may be defined as the distance between the corresponding sections of the meshes. Given a point p and a surface S, the distance e(p,S) is:

$$e(p,S) = \min_{p' \in S} d(p,p') \tag{4.8}$$

where d(•,•) is the Euclidean distance between two points in E^3. The one-sided distance between two surfaces S_1 and S_2 is then defined as:

$$E(S_1,S_2) = \max_{p \in S_1} e(p,S_2) \tag{4.9}$$

A two-sided distance (Hausdorff distance) may be obtained by taking the maximum of $E(S_1, S_2)$ and $E(S_2, S_1)$.

Given a set of uniformly sampled distances, the mean distance E_m between two surfaces is the surface integral of the distance divided by the area of S_1:

$$E_m(S_1,S_2) = \frac{1}{|S_1|}\int_{S_1} e(p,S_2)ds \tag{4.10}$$

The dihedral angle threshold is set to 30 degrees. Each mesh edge with a dihedral angle lower than 30 degrees is classified as a feature edge and its length is summed to the current edge length. Edge length is one of the numerical evaluations in Metro tool.

4.4.2 Data Sets

Two data sets are used to evaluate extended average planes algorithm and other algorithms. Cignoni et al. [1998] collect much data and have an excellent comparison of simplification algorithms. This work works with the same data sets and compare with their data. The first data set is Bunny mesh acquired with an automatic range scanner at Stanford University. It has 34,834 vertices and 69,451 faces. The second data set is from a standard CAD system – Fandisk is a valid representative of CAD models and characterized by sharp edges and sophisticated surface curvature. It has 6,475 vertices and 12,946 faces.

4.4.3 Simplification Algorithms

The developed algorithm is compare with the following simplification algorithms:

1. Mesh Decimation [Schroeder et al., 1992]
2. Simplification Envelopes [Cohen et al., 1996]
3. Multiresolution Decimation [Ciampalini et al., 1997]
4. Mesh Optimization [Hoppe et al., 1993]
5. Progressive Meshes [Hoppe, 1996]
6. Quadric Error Metrics Simplification [Garland and Heckbert, 1997]

Cignoni et al. [1998] run the simplification codes 1,2 and 3 on an SGI Indigo 2 with R 4400 200MHz CPU, 16 KB data cache, 16 KB instruction cache, 1MB secondary cache, and 128 MB RAM. Mesh optimization is executed on a Digital 3000/900 with Alpha 275 MHz CPU and 128 MB RAM. Run times were then scaled back to SGI Indigo 2 units. The progressive Meshes code is executed on a SGI Indigo 2 Extreme with R4400 150 MHz CPU and 128MB RAM. Quadric Error Metric simplification is run on a SGI Indigo 2, R4400 250MHz CPU, 128MB RAM. This Extended average plane algorithm is executed on a Pentium III Personal Computer with 600 MHz CPU and 128MB RAM. Run times are scaled back to SGI Indigo 2 units according to benchmark in the following comparison.

4.4.4 Comparison Evaluation

Tables 4.1-4.2 present the numerical comparison of different simplification algorithms on Bunny mesh and Fandisk mesh. Figures 4.9-4.10 show the maximum and average errors of these algorithms.

Different from Mesh Decimation algorithm, the proposed algorithm applies extended average planes resulting in high accuracy in general. This algorithm simplifies the data set to zero vertex and stores all simplification process. Different from most of the other algorithms [Schroeder et al., 1992; Cohen et al., 1996; Ciampalini et al., 1997; Hoppe et al., 1993; Garland and Heckbert, 1997], this algorithm offers an advantage Progressive Meshes algorithm [Hoppe, 1996] has. It just needs to be executed once to obtain all levels of simplification results.

Compared with the other six algorithms, the maximum error of this algorithm on Bunny mesh is smaller than Mesh Decimation and Quadric Error Metrics. It is smaller than Mesh Decimation, Optimization Mesh and Progressive Mesh on Fandisk mesh. Mesh Decimation causes larger maximum error than this proposed algorithm. The average error of this algorithm is similar to Mesh Optimization, smaller than Progressive Mesh, Quadric Error Metrics and Multiresolution Decimation and much smaller than Simplification Envelope and Mesh Decimation. It is similar to Quadric Error Metrics, smaller than Mesh Optimization and Multiresolution Decimation and much smaller than the others.

This algorithm runs on Pentium III 600 MHz, 128 MB RAM personal computer, the run time in Tables 4.1-4.2 is already scaled to SGI Indigo 2 systems. On bunny mesh, This algorithm runs faster than Mesh Optimization, Progressive Mesh and Simplification Envelopes, similar to Multiresolution Decimation. On Fandisk mesh, only Mesh Decimation costs less time than Extended Average Plane.

The presented algorithm keeps mesh area and volume well on Fandisk mesh. While the performance on mesh area and volume is decreased when the mesh is compressed to

very small face number, for example on bunny mesh. Progressive Mesh is the best on this performance.

According to the above analysis, This algorithm runs fast and has ideal simplification result on maximum and average errors. This work achieves much better simplification results at the sacrifice of negligible computation efficiency comparing with Mesh Decimation. Its performance is similar to Mesh Optimization with improved computational efficiency.

4.5 Summary

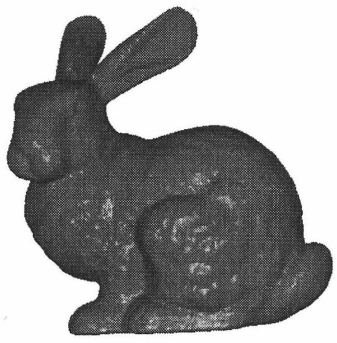
This work has presented an efficient algorithm for solving the mesh simplification problem, approximating an arbitrary mesh by a simplified mesh. The algorithm uses RMS distance error metric to decide the facet curvature. Two vertices of one edge and the surrounding vertices decide the average plane. The distance from the vertices to the average plane is applied to the RMS error measure. In order to raise the importance of the edge vertices in the total error metric, this work applies one weight to the curvature metric that is combined by two edge vertices. The edge length is also considered in the error metric so that the longer edges can be collapsed later and the model geometry can be kept well. Several consistency checks are applied to mesh collapse. The algorithm is tested on one open and two closed meshes. The simplification results are excellent and the computation speed is fast. The algorithm is compared with six other major simplification algorithms using Metro tool. The numerical and graphic comparison results are presented.

Table 4.1 Comparison of simplification algorithms on Bunny Mesh (errors are measured as percentages of the data sets bounding box diagonal; times in seconds).

Bunny (34834 vertices, 69451 triangles, bounding box 15.6x15.4x12.1)						
Edge length 189.099, Area 571.288 (Volume is not defined, the surface is open)						
Mesh Decimation						
N_{Vert}	$N_{triangle}$	E_{max}	E_{avg}	Time	Edgelength	Area
17,566(50%)	34,965	0.1614	0.00735	43.97	194.189	571.457
8,705(25%)	17,267	0.2586	0.01947	37.55	262.279	570.801
3,505(10%)	6,900	0.5212	0.05791	46.22	382.084	568.489
1,775(5%)	3,451	1.2	0.1612	26.68	554.331	563.537
701(2%)	1,389	2.0721	0.3423	35.22	572.173	555.25
Simplification Envelope						
17,418(50%)	34,643	0.02089	0.00414	932.17	322.801	571.316
8,709(25%)	17,252	0.04154	0.01174	942.93	299.725	570.91
3,472(10%)	6,801	0.08813	0.03117	944.66	396.175	570.962
1,763(5%)	3,395	0.1629	0.06066	964.03	478.302	569.133
678(2%)	1,301	0.3802	0.1423	1003.01	468.944	562.575
Multiresolution Decimation (Jade 2.0)						
17,417(50%)	34,679	0.0224	0.0036	208.95	207.26	571.591
8,708(25%)	17,289	0.0438	0.0097	371.03	218.088	571.537
3,483(10%)	6,874	0.0948	0.0242	388.68	272.793	571.145
1,741(5%)	3,408	0.1697	0.0459	438.62	361.983	570.392
696(2%)	1,358	0.3519	0.0997	475.73	416.856	567.973
Mesh Optimization						
17,410(50%)	34,643	0.2968	0.00996	7,100	1346.74	579.836
8,699(25%)	17,279	0.03668	0.01064	7,400	488.486	576.048
3,501(10%)	6,956	0.2964	0.01554	7,600	358.65	577.56
1,758(5%)	3,491	0.3666	0.02415	7,400	364.89	581.119
686(2%)	1,347	0.4262	0.04846	8,000	392.326	585.416
Progressive Meshes						
17,417(50%)	34,667	0.1339	0.00781	-	255.977	571.704
8,708(25%)	17,252	0.1585	0.011	-	283.746	572.252
3,483(10%)	6,821	0.2065	0.01851	-	327.247	573.149
1,741(5%)	3,367	0.2817	0.03273	-	391.144	573.949
696(2%)	1,359	0.529	0.06897	1,450	451.311	574.997
Quadric Error Metrics						
17,417(50%)	34,666	1.024	0.00486	20.5	203.919	568.542
8,708(25%)	17,293	1.6769	0.01135	24.7	223.754	567.082
3,483(10%)	6,888	1.707	0.02363	26.42	259.867	565.728
1,741(5%)	3,434	2.4256	0.04279	26.93	360.545	563.636
696(2%)	1,360	4.27	0.1031	27.36	429.64	558.917
Average Planes						
17,417(50%)	34,687	0.2495	0.002206	-	411.371	572.16
8,708(25%)	17,304	0.3267	0.005348	-	609.79	571.998
3,483(10%)	6,894	0.5686	0.01285	-	563.623	566.983
1,741(5%)	3,429	0.6355	0.02396	-	572.376	562.612
696(2%)	1,360	0.7603	0.05189	564	530.526	551.351

Table 4.2 Comparison of simplification algorithms on Fandisk Mesh (errors are measured as percentages of the data sets bounding box diagonal; times in seconds).

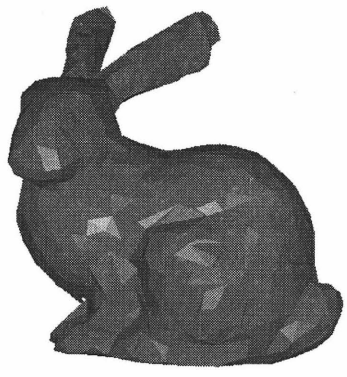
Fandisk (6,475 vertices, 12,946 triangles, bounding box 4.8x5.6x2.7)							
Edge length 69.9526, Area 60.6691, Volume 20.2433							
Mesh Decimation							
N_{vert}	$N_{triangle}$	E_{max}	E_{avg}	Time	Edgelength	Area	Volume
3,224(50%)	6,444	0.00412	4.50E-05	4.5	69.9526	60.6691	20.2432
1,616(25%)	3,228	0.07452	0.00398	7.38	69.95	60.6667	20.2557
639(10%)	1,274	0.2471	0.01539	10	73.8872	60.6585	20.265
325(5%)	646	1.1708	0.058	7.62	79.8004	60.64	20.3816
Simplification Envelopes							
3,216(50%)	6,428	0.00317	0.000158	203.49	87.0931	60.6691	20.2432
1,633(25%)	3,262	0.02227	0.002505	228.14	78.6976	60.6732	20.2466
654(10%)	1,304	0.05958	0.009646	185.22	70.3327	60.6747	20.2731
317(5%)	630	0.1468	0.02452	170.93	72.7624	60.6733	20.3143
Multiresolution Decimation (Jade 2.0)							
3,149(50%)	6,294	0.00248	4.85E-05	28.26	69.9526	60.6691	20.2433
1,615(25%)	3,226	0.00427	0.00021	37.72	69.9526	60.6694	20.2433
645(10%)	1,286	0.02657	0.0028	52.26	70.5093	60.6716	20.2497
323(5%)	642	0.06778	0.00944	64.79	70.2319	60.6814	20.2686
Mesh Optimization							
3,287(50%)	6,570	0.5297	0.002776	2,000	112.087	60.8125	20.2395
1,611(25%)	3,218	0.5021	0.002901	2,100	89.7844	60.8107	20.249
655(10%)	1,306	0.2452	0.003614	2,300	73.9007	60.7556	20.2412
333(5%)	662	0.291	0.005877	2,500	72.4195	60.7721	20.2435
Progressive Meshes							
3237(50%)	6,470	0.1366	0.003451	-	70.6122	60.6412	20.241
1,618(25%)	3,232	0.1674	0.004139	-	71.1858	60.6515	20.2401
647(10%)	1,290	0.2377	0.006077	-	72.7364	60.6812	20242
323(5%)	642	0.247	0.01186	285	79.7237	60.7464	20.2522
Quadric Error Metrics							
3237(50%)	6,470	0.00067	8.62E-05	42.74	87.6357	61.3195	20.2433
1,618(25%)	3,232	0.00442	0.00021	44.41	79.0333	61.1082	20.2434
647(10%)	1,290	0.03746	0.00183	45.11	76.4246	61.1372	202456
323(5%)	642	0.09253	0.00581	47.63	74.6483	61.0833	20.2537
Average Planes							
3237(50%)	6,470	0.001482	1.56E-05	-	87.0174	60.8172	20.2433
1,618(25%)	3,232	0.04331	0.000521	-	93.432	60.993	20.263
647(10%)	1,290	0.08747	0.00299	-	143.273	61.0285	20.272
323(5%)	642	0.1418	0.00681	37.9	125.968	60.9144	20.256



69,451 faces, 3.392 MB

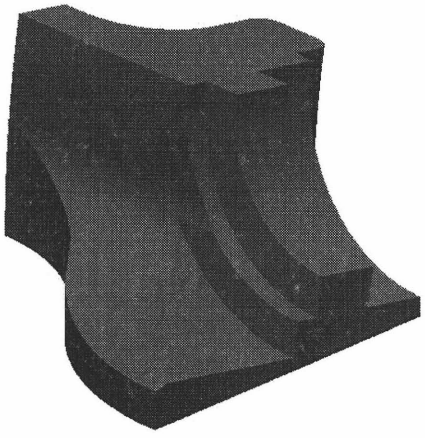
17,304 faces, 846 KB

6,894 faces, 337 KB

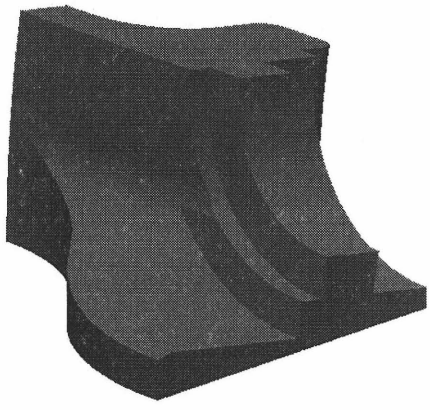


1,360 faces, 67 KB

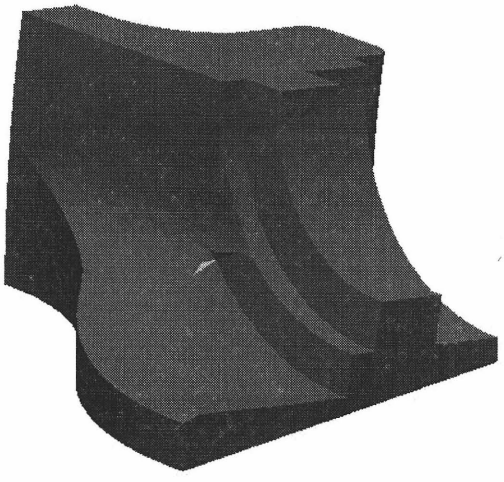
Figure 4.6 The shaded original and simplified bunny models.



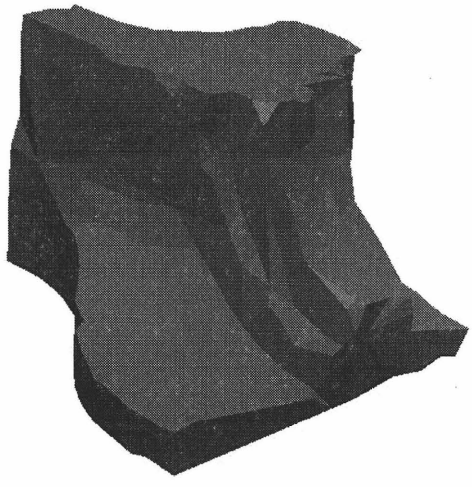
6,475 faces, 633 KB



3,237 faces, 316 KB



1,618 faces, 158 KB



323 faces, 32 KB

Figure 4.7 The fandisk model simplification results.

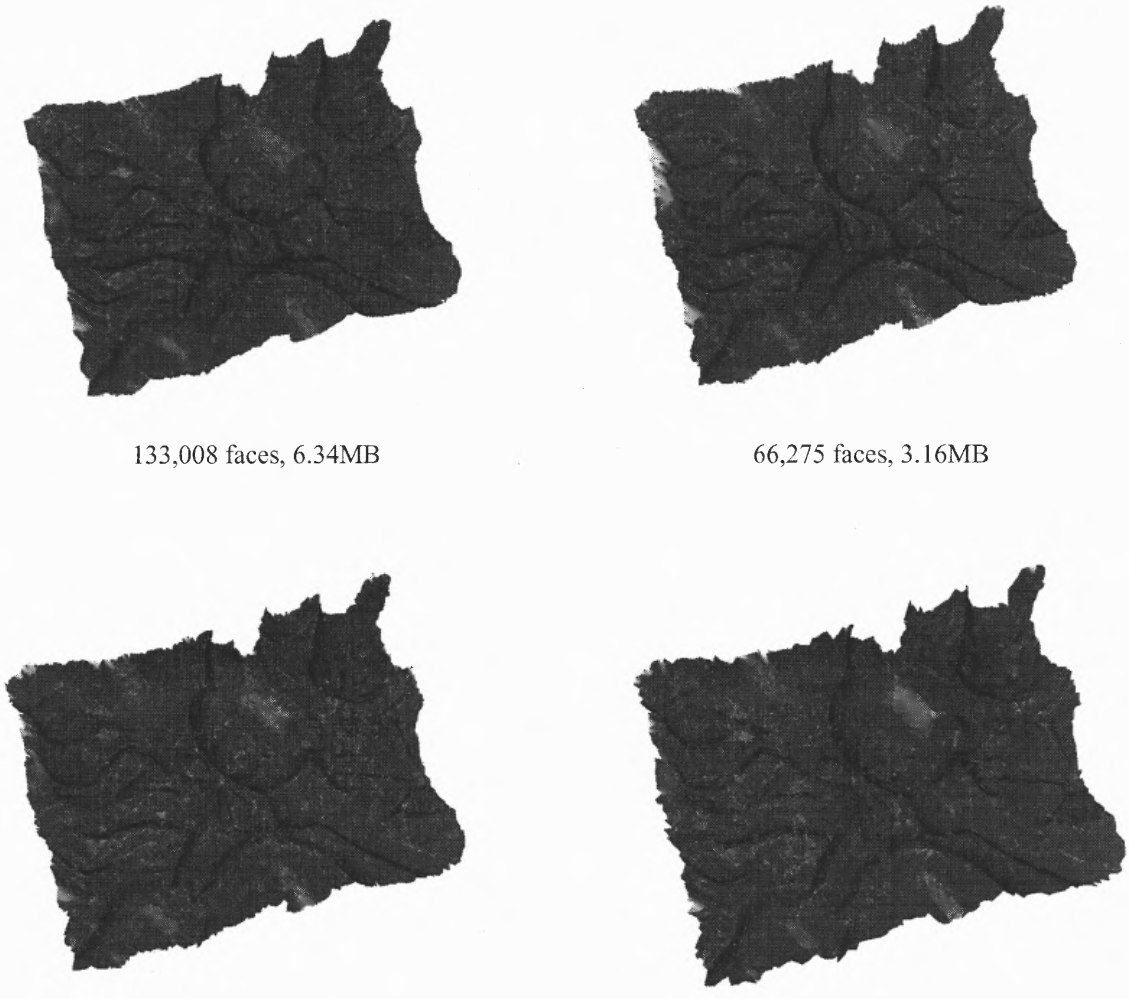
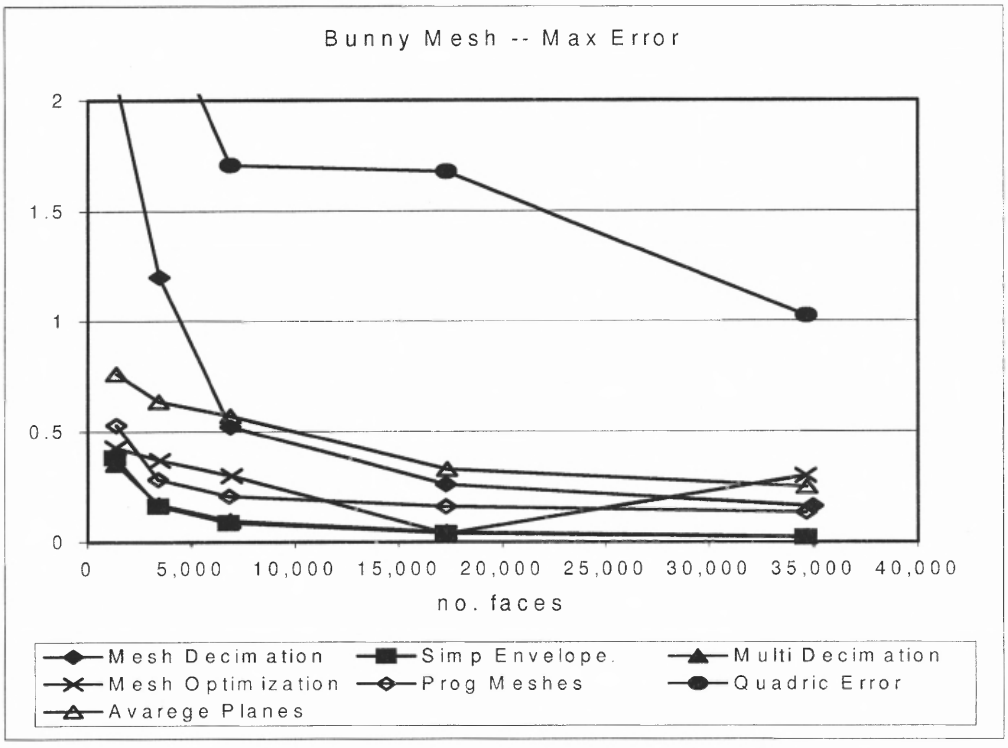


Figure 4.8 The ChristmasMan model simplification results.



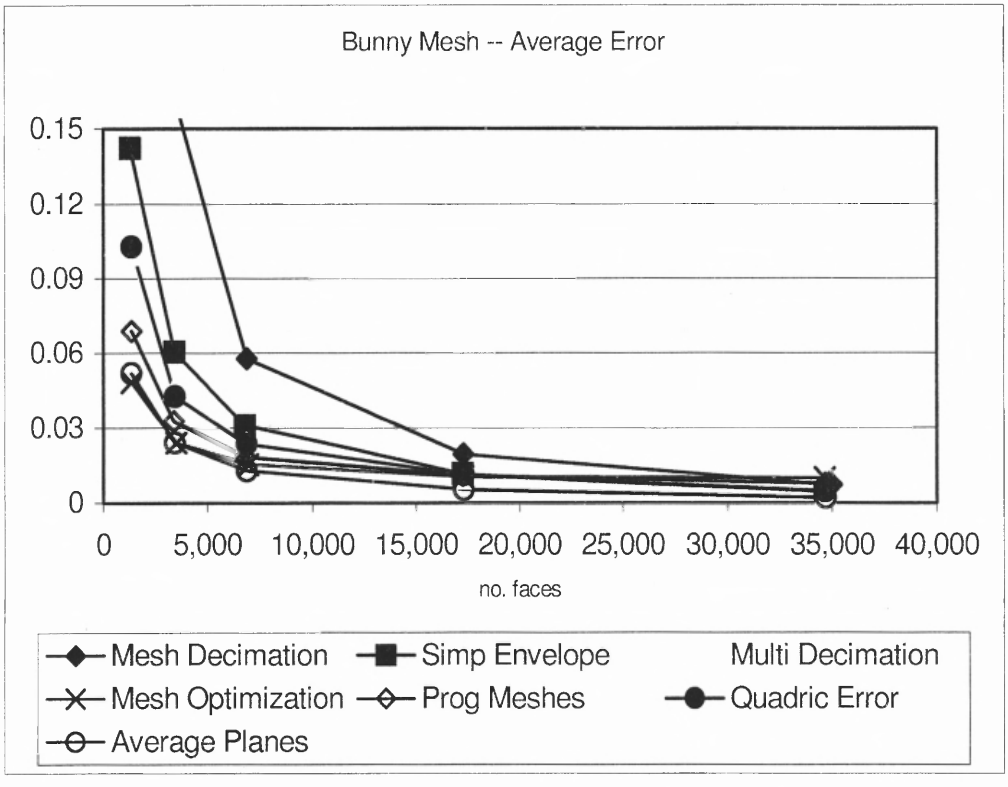


Figure 4.9 The performance of various simplification algorithms on Bunny Mesh.

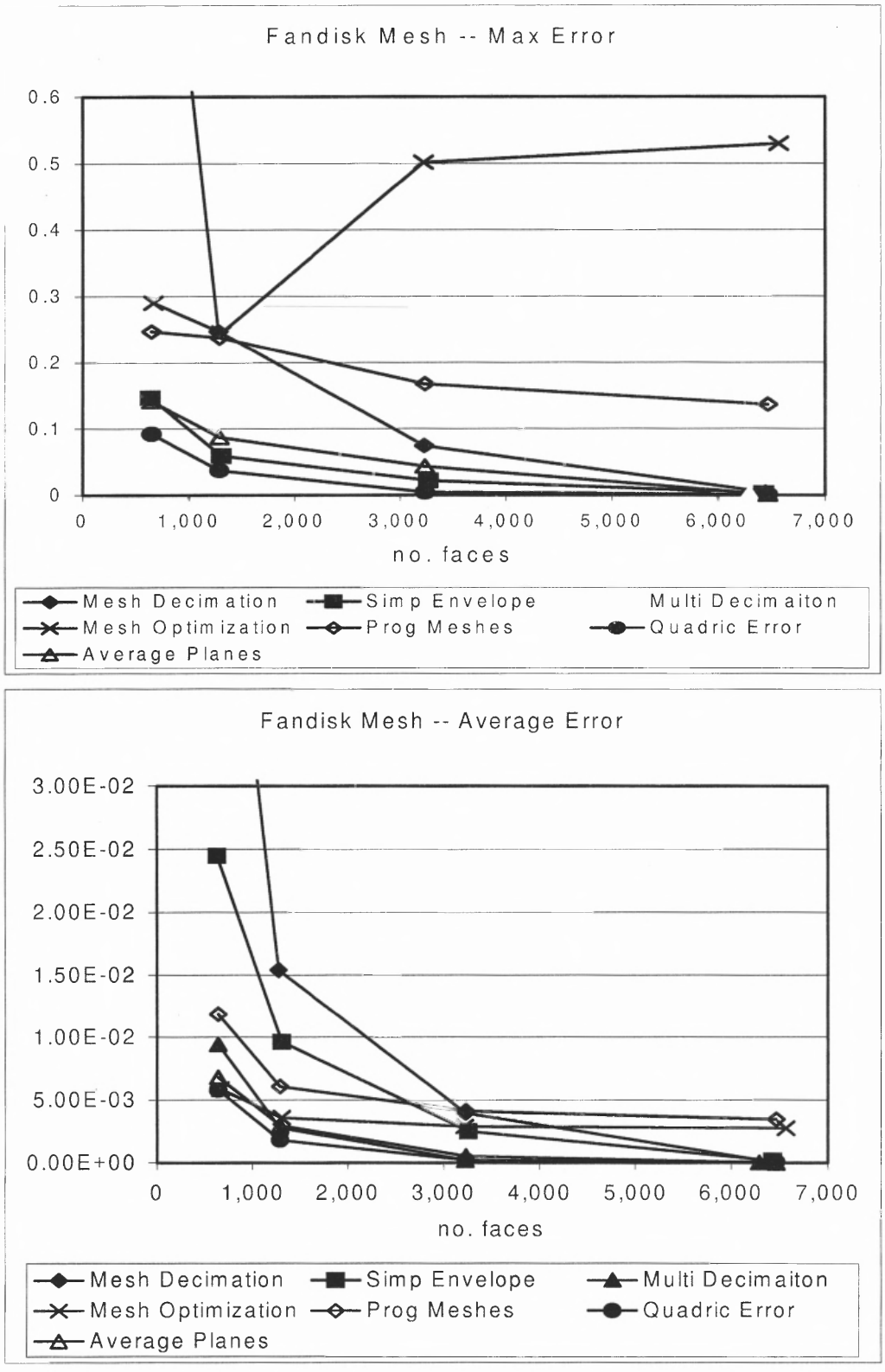


Figure 4.10 The performance of various simplification algorithms on Fandisk Mesh.

CHAPTER 5

IMAGE WARPING AND MORPHING

5.1 Introduction

Image warping and morphing has received much attention in recent years. It has proven to be a powerful tool for visual effects, enabling the fluid transformation of one digital image into another.

Advances in 3D scanning and acquisition technology have made dense triangle meshes popular as representations of complex objects. These scanning devices typically create triangulations that form a surface of arbitrary topology. If there are two volumetric models, the problem of transition from one geometrical model to another can be formally stated as follows: given two models, a source S and a target T, construct a set of transformations $\{W_t \mid t \in [0,1]\}$, such that $W_0 = I$, the identity transformation, or $W_0(S) = S$, and $W_1(S) = T$. Intermediate models or in-between "blends" of S and T are defined by $W_t(S)$, $t \in (0,1)$.

Image morphing describes methods for deforming images to arbitrary shapes. Digital image morphing received much interest in the recent years, mainly due to a large range of applications. Some applications are in "artistic" areas, like computer animation in movies, games and advertising industries. Image warping (morphing) is a basis for two-dimensional morphing, the smooth transition between keyframe images. Other artistic applications include facial animation and free-form deformation of images. Other applications are in scientific image processing. In image data acquisition, an acquisition method often deforms the acquired image, for example, through lens distortion. In satellite imaging, distortions are caused by surface curvature and oblique viewing angles.

In ultrasonic medical imaging, distortions are caused by the varying speed of sound in different materials. These deformations must be rectified to obtain the correct coordinates in a registration process. Another important application is in a medical image field. Image morphing technique can help doctors to perform better nose and breast surgery.

The term "morphing" is used for all methods that gradually and continuously deform S into T, while producing the in-between models. The morphing process is usually composed of warping and interpolation. Since morphing involves two objects, warping operation is performed first, to obtain an approximated alignment so that the two shapes will be relatively similar, and then the two warped objects are blended into one. This interpolation process in image-based techniques is often referred to as cross-dissolving, where the values of corresponding pixels are blended, yielding the effect of fading-in of the target image with a fade-out of the source image.

If $\vartheta_1 = (U_1, f_1)$ and $\vartheta_2 = (U_2, f_2)$ are two graphical objects, with $U_1, U_2 \subset R^m$, a morphing or metamorphosis between ϑ_1 and ϑ_2 is a k-parameter continuous family of transformations:

$$F: \vartheta_1 \times R^k \rightarrow R^n \tag{5.1}$$

such that there are parameter values v_0 and v_1 satisfying $F_{v0}(\vartheta_1) = \vartheta_1$ and $F_{v1}(\vartheta_2) = \vartheta_2$. Intuitively, for each parameter $v \in R^k$ from the parameter space, a new graphical object $\vartheta_v = F_v(\vartheta_1)$ is obtained, and this family ϑ_v, $v \in R^k$, performs the transition from one object to the other, as v varies in the parameter space. Thus, a metamorphosis can be interpreted as a continuous deformation from object ϑ_1 to object ϑ_2. The object ϑ_1 is called the source of the metamorphosis, and the object ϑ_2 is called the target.

Warping can be generally described as a mapping or transformation that can be applied either in image or object space. Consider a graphical object $\vartheta = (U, f)$, $U \subset R^m$. A continuous k-parameter family of transformations $F: \vartheta \times R^k \rightarrow R^n$ is warping of the graphical object ϑ. Intuitively, a warping is a continuous deformation of a graphical object. For each fixed vector $v \in R^k$, the graphical object $F_v(\vartheta)$ is called an instantiation of the warping. The mapping essentially distorts the object it is applied to, according to some predetermined constraints, which are usually specified by the user. Such user control over the warping is crucial in obtaining satisfactory results, and can be achieved, for instance, through specifying a predetermined mapping for a set of points (point-to-point correspondence). In practice, warping (and hence morphing) requires a good level of user specification to achieve the desired results. All forms of specification (e.g., point-to-point correspondence) are based on the idea of defining the transformation by manipulating only a finite and small number of parameters. From these specifications, mathematical techniques should be used to compute the transformation for any point in the warping domain [Cohen-Or et al., 1998]. The key to a successful morphing method is its ability to achieve aesthetically pleasing morphs with few such feature point pairs while providing means for very detailed control if so desired.

5.2 2D Image Warping and Morphing

Research effort in the morphing field initially focused on the 2-D problem. Sederberg and Greenwood [1992] propose a solution to the vertex correspondence problem, and the vertex path problem is tackled by [Sederberg et al., 1993]. Ruprecht and Muller [1995]

propose one approach to image warping based on scattered data interpolation methods to provide smooth deformations with easily controllable behavior. Guibas and Hershberger [1994] investigate the problem of morphing one simple polygon into another. They assume that two initial polygons have the same number of sides n, and that corresponding sides are parallel. A morph is possible by a varying simple interpolating polygon also of n sides parallel to those of the two original ones. A treatise of image warping methods can be found in Wolberg's 1990 book. Wolberg [1998] also surveys the growth of this field and describes recent advances in image morphing in terms of feature specification, warp generation methods, and transition control. Several morphing algorithms, including those based on mesh warping, field morphing, radial basis functions, thin plate splines, energy minimization, and multilevel free-form deformations were reviewed in [Wolberg, 1998].

Mesh warping: Mesh warping was pioneered at industrial Light & Magic (ILM) by D. Smythe for use in the movie Willow in 1988 [Smythe, 1990]. It has been successfully used in many subsequent motion pictures. Suppose I_s and I_t are the source and target images, respectively. The source image is associated with mesh M_s. It specifies the coordinates of control points, or landmarks. A second mesh M_T specifies their corresponding positions in the target image. All intermediate frames in the morph sequence are the product of a four-step process:

for each frame *f* **do**

Linearly interpolate mesh M, between M_S and M_T
Warp I_S to I_1, using meshes M_S and M
Warp I_T to I_2, using meshes M_T and M
Linearly interpolate image I_f, between I_1 and I_2

end

This process demonstrates that morphing is simply a cross-dissolve applied to warped imagery.

Field morphing: Beier and Neely [1992] suggest an excellent algorithm based on the fields of influence surrounding 2D control primitives. A pair of lines (one defined relative to the source image, and the other defined relative to the destination image) defines a mapping form one image to the other.

If the transformation is between one pair of lines, this pair of corresponding lines in the source and destination images define a coordinate mapping from the destination image pixel coordinate X to the source X' such that for a line PQ in the destination image and P'Q' in the source image

$$u = \frac{(X-P).(Q-P)}{\|Q-P\|^2} \tag{5.2}$$

$$v = \frac{(X-P).Perpendicular(Q-P)}{\|Q-P\|} \tag{5.3}$$

$$X' = P' + u.(Q'-P') + \frac{v.Perpendicular(Q'-P')}{\|Q'-P'\|} \tag{5.4}$$

where *Perpendicular*(·) returns the vector perpendicular to, and the same length as, the input vector. There are two perpendicular vectors. Either the left or right one can be used, as long as it is consistently used throughout.

The value u is the position along the line, and v is the distance from the line. The value u goes from 0 to 1 as the pixel moves from P to Q. u is less than 0 or greater than 1 if the pixel moves outside that range. The value for v is the perpendicular distance in pixels from the line. If there is just one line pair, the transformation of the image proceeds as follows:

For each pixel X in the destination image: find the corresponding *u* and *v*; find the X' in the source image for that *u* and *v*; destinationImage (X) = sourceImage (X').

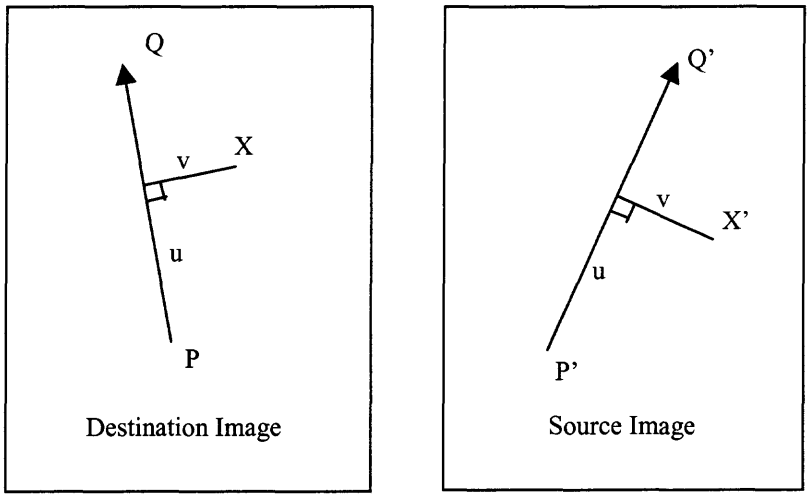


Figure 5.1 Single line pair.

In Figure 5.1, X' is the location to sample the source image for the pixel at X in the destination image. The location is at a distance *v* (the distance from the line to the pixel in the source image) from the line P'Q', and at a proportion of *u* along that line. The algorithm transforms each pixel coordinate by a rotation, translation, and/or a scale, thereby transforming the whole image.

The upper left of Figure 5.2 shows the original image. The line is rotated in the upper right image, translated into the lower left image, and scaled in the lower right image, performing the corresponding transformations to the image.

If the transformation is between multiple pairs of lines, these multiple pairs of lines specify more complex transformations. A weighting of the coordinate transformations for each line is performed. A position X_i' is calculated for each pair of lines. The

displacement $D_i = X_i' - X$ is the difference between the pixel locations in the source and destination images, and a weight average of those displacements is calculated. The weight is determined by the distance from X to the line. This average displacement is added to the current pixel location X to determine the position X' to sample in the source image. The single line case falls out as a special case of the multiple line case. It is assumed that the weight never goes to zero anywhere in the image. The weight assigned

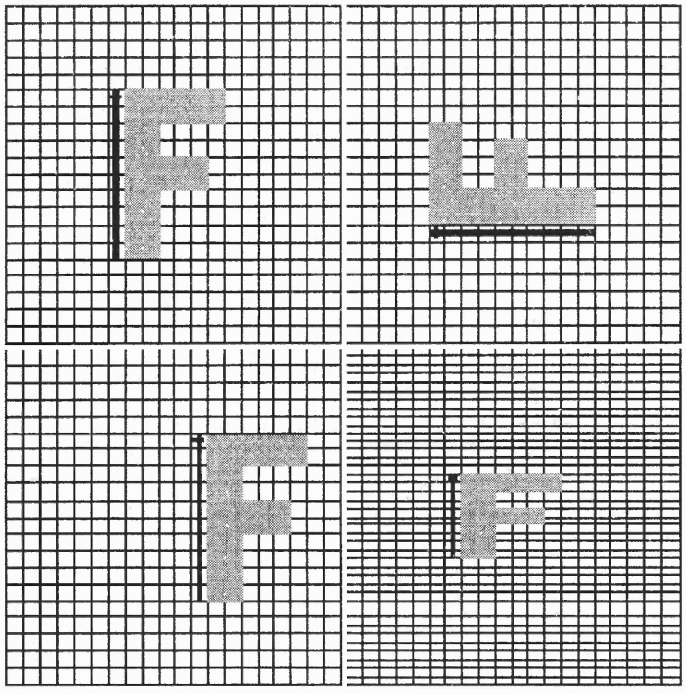


Figure 5.2 Single line pair example [Beier and Neely, 1992].

to each line should be the strongest when the pixel is exactly on the line, and weaker the further the pixel is from it. The equation is:

$$Weight = \left(\frac{length^p}{a + dist} \right)^b \tag{5.5}$$

where *length* is the length of a line, *dist* is the distance from the pixel to the line, and *a, b,* and *p* are constants that can be used to change the relative effect of the lines (Figure 5.3).

In Figure 5.3, X' is the location to sample the source image for the pixel at X in the destination image. That location is a weighted average of the two pixel locations X_1' and X_2', computed with respect to the first and second line pair, respectively.

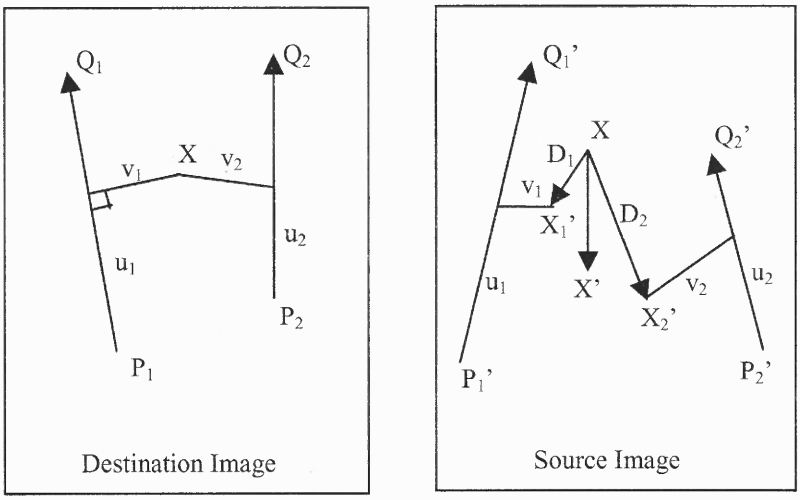


Figure 5.3 Multiple line pairs.

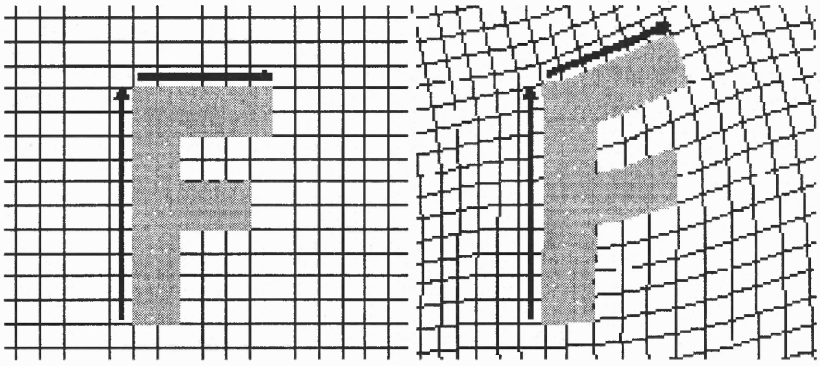


Figure 5.4 Multiple line pair example [Beier and Neely, 1992].

With two or more lines, the transformation is not simple. In Figure 5.4, the figure on the left is the original image. It is distorted by rotating the line above F around its first point. The whole image is distorted by this transformation. It is still not possible to do a uniform scale or a shear with multiple lines. Almost any pair of lines results in a non-affine transformation. Still, it is fairly obvious to the user what happens when lines are added and moved. Pixels near the lines are moved along with the lines, pixels equally far away from two lines are influenced by both of them.

Although this approach simplifies the specification of feature correspondence, it complicates warp generation. This is due to the fact that all line pairs must be considered before the mapping of each source point is known. This global algorithm is slower than mesh warping, which uses bicubic interpolation to determine the mapping of all points not lying on the mesh.

Radial basis functions: The most general form of feature specification permits the feature primitives to consist of points, lines, and curves. Since lines and curves can be point sampled, it is sufficient to consider the features on an image to be specified by a set of points. In that case, the x and y components of a warp can be derived by constructing the surfaces that interpolate scattered points. Consider, for example, M feature points labeled (u_k, v_k) in the source image and (x_k, y_k) in the target image, where $1 \leq k \leq M$. This formulation permits us to draw upon a large body of work on scattered data interpolation to address the warp generation problem. All subsequent morphing algorithms have facilitated general feature specification by appealing to scattered data interpolation. Warp generation by this approach is extensively surveyed by [Ruprecht and Muller, 1995; Wolberg, 1990].

Energy minimization: All of the methods just described do not guarantee the one-to-one property of the generated warp functions. When a warp is applied to an image, the one-to-one property prevents the warped image from folding back upon itself. Lee et al. [1996] propose an energy minimization method for deriving one-to-one warp functions. Their method allows extensive feature specification primitives such as points, polylines, and curves. Internally, all primitives are sampled and reduced to a collection of points. These points are then used to generate a warp that is interpreted as a 2D deformation of a rectangular plate. A deformation technique is provided to derive C^1-continuous and one-to-one warps from the positional constraints. The requirements for a warp are represented by energy terms and satisfied by minimizing their sum. The technique generates natural warps since it is based on physically meaningful energy terms. The performance of the method, however, is hampered by its high computational cost.

Multilevel free-form deformation: Lee et al. [1995] present a new warp generation method that is much simpler and faster than the related energy minimization method [Lee et al., 1996]. Large performance gains are achieved by applying multilevel free-form deformation (MFFD) across a hierarchy of control lattices to generate one-to-one and C^2 continuous warp function. In particular, warps are derived from positional constraints by introducing the MFFD as an extension to free-form deformation (FFD) [Sederberg and Parry, 1986]. In their paper, the bivariate cubic B-spline tensor product is used to define the FFD function. A new direct manipulation technique for FFDs, based on 2D B-spline approximation, is applied to a hierarchy of control lattices to exactly satisfy the positional constraints. To guarantee the one-to-one property of a warp, a sufficient condition for a 2D cubic B-spline surface to be one-to-one is presented. The MFFD generates C^2-

continuous and one-to-one warps that yield fluid image distortions. The MFFD algorithm is combined with the energy minimization method [Lee at al., 1996] in a hybrid approach.

Figure 5.5 gives an example in which the MFFD is applied to generate a deformation of plate W from positional constraints. Figure 5.5 (a) shows the selected points in the undeformed shape of the plate. Figure 5.5 (b) through (e) show a sequence of deformations in which the deformed positions of the selected points gradually approach the specified positions. Figure 5.5 (f) shows the resulting deformation with the specified positions. In this example, the FFD manipulations are performed no more than twice at each level of the control lattice. The benefit of this approach of this approach is that feature specification is more expressive and less cumbersome. Rather than editing a mesh, a small set of feature primitives are specified. To further assist the user, snakes are introduced to reduce the burden of feature specification. Snakes are energy-minimization splines that move under the influence of image and constraint forces. Snakes streamline feature specification because primitives must only be positioned near the features. Image forces push snakes toward salient edges, thereby refining their final positions and making it possible to capture the exact position of a feature easily and precisely.

Work minimization: Given that two images are sufficiently similar, image registration techniques [Brown, 1992] offer a potential solution for automatically determining the correspondence among all points across the source and target images. Such methods exploit the fact that the two frames represent the same scene imaged only moments apart, and therefore only small displacements must be recovered. This is

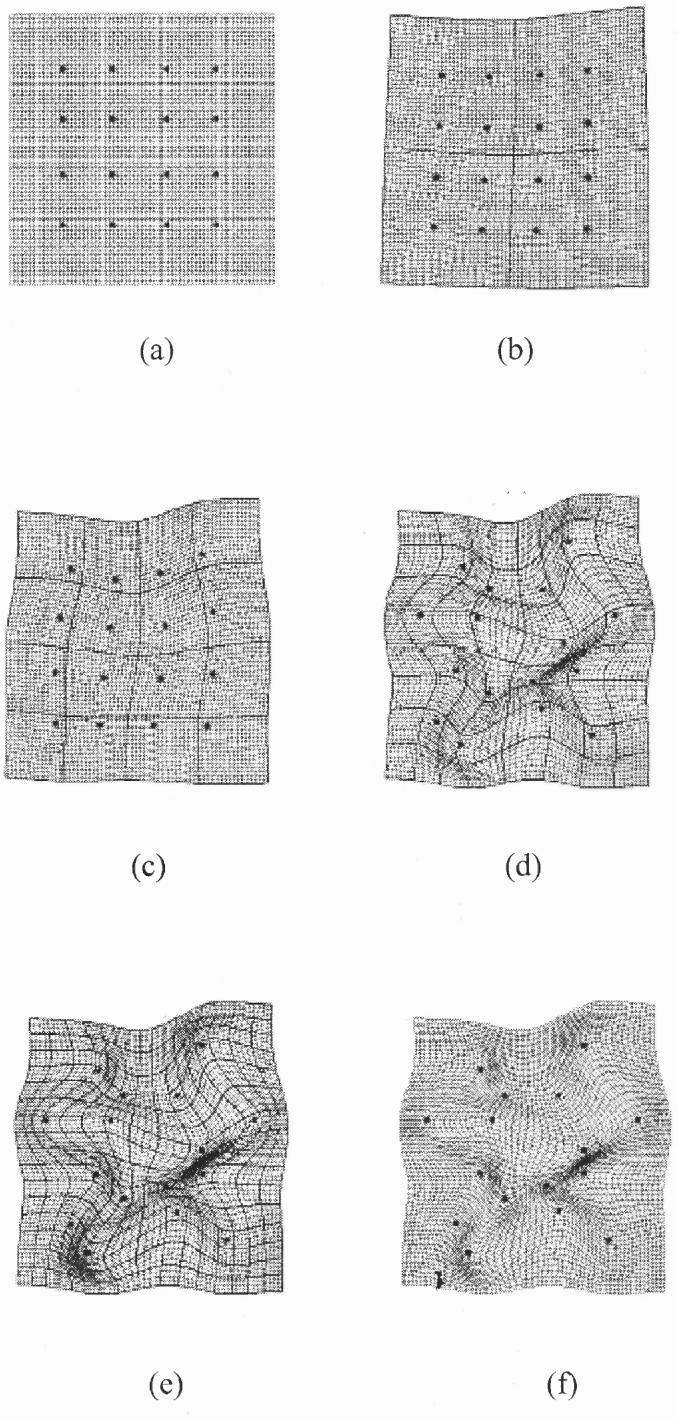


Figure 5.5 An example of the MFFD [Lee et al., 1995].

generally not true in morphing applications, where the source and target images may be vastly different. Nevertheless, it is reasonable to consider the extent to which the feature specification problem can be automated for similar input images. This is precisely the problem considered by [Gao and Sederberg, 1998].

Gao and Sedergberg [1998] present a new algorithm for determining the warp function between two similar images with little or no human interaction. This approach is consistent with the trend towards minimizing the amount of user interaction in specifying feature correspondence. The algorithm is based on a work minimization approach that derives its cost directly from the image intensities, and not from user-specified constraints. Motivated from their previous work for solving a polygon shape blending problem [Sederberg and Greenwood, 1992], they attempt to minimize the cost associated with warping and recoloring the image. They apply a hierarchical warp using a grid of bilinear B-spline patches and associate a cost function to it. Even after the warp is applied, there will be differences in the intensity values and a cost function is thus associated with recoloration. The morph generation process is then reduced to that of finding the global minimum cost function among all possible warps. The authors offer a method that is simple and fast, although it does not guarantee a global minimum.

The premise of the work minimization algorithm is that the most visually pleasing morphs are associated with those images that satisfy the minimal work constraints. This assumption is sometimes false, and therefore some user interaction is necessary to specify features and initialize a warp. In spite of any initial warp that may need to be specified, the rationale behind this approach is that user interaction can be significantly reduced by

exploiting work minimization constraints. Although this work is still in its early stages, it has shown promising results for morphing among similar images.

5.3 3D Image Warping and Morphing

Less work has been done in 3D morphing. Lazarus and Verroust [1998] give an excellent survey of previous work on 3D morphing problem. They specify that there are an unlimited number of ways to interpolate from one object to another. Such interpolations may be performed for geometry as well as attributes such as color.

Algorithms for morphing are evaluated mainly by criteria related to the ease with which the results can be controlled and the aesthetic quality of the results themselves. Ease encompasses both the amount of work an artist has to invest, as well as the predictability of the result. Since aesthetic quality is subjective, precise user control is important.

3D morphing overcomes the following shortcomings of 2D morphing as applied to images generated from 3D models [Lerios et al., 1995]:

(1) In 3D morphing, creating the morphs is independent of the viewing and lighting parameters. Hence, a morph sequence can be created once, and then experiment with various camera angles and lighting conditions during rendering. In 2D morphing, a new morph must be recomputed every time people wish to alter their viewpoint or the illumination of the 3D model.

(2) 2D techniques, lacking information on the model's spatial configuration, are unable to correctly handle changes in illumination and visibility. Two examples of this type of artifact are: 1) Shadows and highlights fail to match shape changes occuring in the

morph, and 2) when a feature of the 3D object is not visible in the original 2D image, this feature cannot be made to appear during the morph; for example, when a singing actor needs to open his mouth during a morph, pulling his lips apart thickens the lips instead of revealing his teeth.

Most methods for morphing 3D objects use either discrete or combinatoric representations of the objects themselves. Discrete representations typically voxelize objects or their distance functions and aim to extend 2D morphing algorithms to 3D [Beier and Neely, 1992; Lee et al., 1995].

There are several ways to represent an object in a computer. The object representations and their corresponding data structures usually have a strong impact on the type and difficulty of algorithms involved in transforming one object into another. At a very coarse level, there are three kinds of shape representations [Lazarus and Verroust, 1998]:

(1) Objects can be described as level sets of functions defined on the whole 3D space. Implicit surfaces and voxelized objects fall into this category. The voxelized objects may indeed be considered as a level set of its characteristic functions with a value of one at the object voxels and zero elsewhere. One can also interpret a voxelized object as the zero level set of the discrete distance to the object voxels. If an object is defined by the set of points p such that $f(p) = c$ for some function f and level c, it is easy to define the interior of this object by considering all the points p such that $f(p) \leq c$. For this reason people qualify morphing techniques based on this category of objects as volume based ones;

(2) Objects, such as terrains, can be represented as an elevation map over a planar domain; and

(3) Objects can be represented by their boundaries as a 2D surface, such as a polyhedral surface or a spline surface.

5.3.1 Volume-Based Approaches

The first category of objects is often preferred for morphing application [Lerios et al., 1995; Hughes, 1992; He et al., 1994; Cohen-Or et al., 1998]. There are few restrictions on the form of the functions defining the level sets. It follows that any kind of continuous interpolation between the functions that define the source object and the target object produces at least some smooth transformation. If the source object is expressed by the set of points p such that $f_0(p) = c_0$, and the target object is given by $f_1(p) = c_1$, an interpolated object for each $t \in [0, 1]$ can be defined with the following equation:

$$(1-t)\, f_0(p) + t\, f_1(p) = (1-t)\, c_0 + t\, c_1 \qquad (5.6)$$

Hughes [1992] and He et al. [1994] propose methods working in the Fourier domain. This provides novel controls over the morph by treating individual frequency bands with different functions of time. The coefficients of the decomposition are interpolated in order to define the coefficients of the interpolated object function. The interpolated object function is further recomposed with these coefficients.

Cohen-Or et al. [1998] present an object-space metamorphosis method between two (or more) surfaces of solid objects of general topology, using an extended distance field interpolation. The metamorphosis of the surface is guided by corresponding control points that define a warp function, which is decomposed into a rigid part and an elastic part in order to reduce the distortion of intermediate models. The technique has two steps: warp and interpolation step. The warp step is derived from the matching of two sets of

feature points. The warp is decomposed into a rigid transformation followed by a small perturbation expressed in terms of radial functions. The interpolation step is reduced to a linear interpolation of distances fields deformed by the warp. The distance fields are computed from the voxelized objects with a 3D distance transform. Figure 5.6 shows a 3D morphing sequence between the surfaces of a Triceratops and an iron, is controlled by 16 anchor points using the Gaussian as a radial function. Only the iron object has a hole [Cohen-Or et al., 1998].

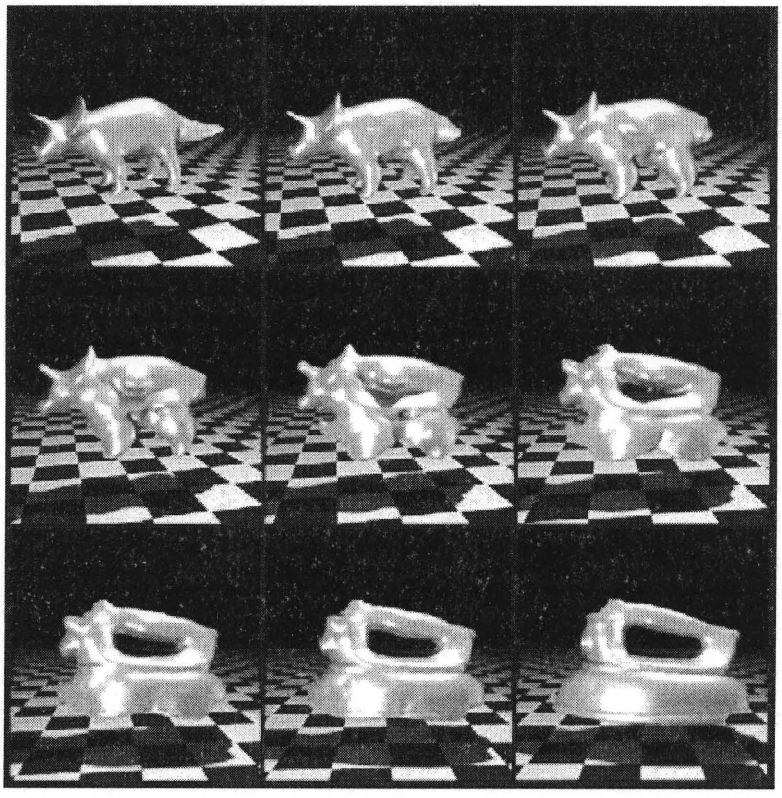


Figure 5.6 The metamorphosis of a triceratops into an iron [Cohen-Or et al., 1998].

Blanz and Vetter [1999] introduce a new technique for modeling textured 3D faces. 3D faces can either be generated automatically from one or more photographs, or modeled directly through an intuitive user interface.

Lerios et al. [1995] extend the work [Beier and Neely, 1992] and use the fields of influence of 3D primitives to warp volumes. They also create each morph in two steps, warping and blending. In the warping step, source volume S and target volume T are warped to obtain volumes S' and T'. Their warping technique allows the animator to define quickly the exact shape of objects represented in S' and T', thus meeting the realism condition. In the blending step, S' and T' are combined into one volume, the morph. Their blending technique provides users with sufficient control to create a smooth morph.

Lee et al. [1999] present a new method for user controlled morphing of two homeomorphic triangle meshes of arbitrary topology. They address the problem of establishing dense correspondences between any two irregular connectivity meshes with the only requirement that they be topologically equivalent. This involves the construction of mappings from the fine meshes to their coarse base domains and of a mapping between the base domains. These mappings are realized through the metamesh, a topologically and geometrically merged version of source and destination meshes. The necessary computations are efficient enough to allow us to compute high quality morphs on meshes with thousands of triangles on a low end PC within several minutes. They provide easy controls for the mapping from source to destination. In the case of fine features, such as vertices or connected sets of edges, simply marking them on each mesh and pairing them up is sufficient. Coarse control can be exercised by interactively

modifying the mapping between the coarse source and destination domains. Providing a small set of feature pairs is generally sufficient to achieve aesthetically pleasing results.

Their algorithm proceeds by first creating parameterizations of the source and destination mesh using the Multi-resolution Adaptive Parameterization of Surface (MAPS) algorithm of [Lee et al., 1998]. MAPS controls the parameterization using as few or as many features on the original meshes as that users desire. These two parameterizations are then put into correspondence through the construction of a map between the source and destination domains. This stage provides additional controls to the user to influence the morph in a broad fashion. The composition of these stages is used for subsequent shape interpolation. The main restriction of their method is the requirement that source and target share the same genus. Thus fundamental work on extending MAPS to deal with genus changes is still needed.

5.3.2 Approaches Based on Boundary Representations

Boundary representations are very popular for representing 3D objects and 3D virtual worlds. A large number of models and data structures have been proposed to represent objects by their boundaries. The polygonal surfaces and parameterized surfaces, such as spline surfaces, are the two main models used in the graphics community. The corresponding data structures usually contain topology and geometry information. The topology gives the adjacency relationship between polygonal faces or parameterized patches, while the geometry describes the precise coordinates of the vertices or control points that define faces or patches [Lazarus and Verroust, 1998]. The use of boundary representations has several advantages. The corresponding data structures are more

compact than the storage of voxelized objects. Many practical and efficient algorithms are available to visualize objects represented by their boundaries. It is also very easy to attach properties such as color, normal, or texture to boundary representation. The presence of topology and geometry in a boundary representation generally leads to splitting the morphing problem into two steps: establishing a correspondence between the source and target object and interpolating the positions or geometry of the corresponding features. The correspondence step aims at constructing a single mesh with two geometric instantiations: one for each source and target object. This single mesh can be obtained by merging the two object meshes as in [Kent et al., 1992], or by creating a new common mesh as in [Lazarus and Verroust, 1997]. The correspondence problem remains a difficult step.

Kent et al. [1992] offer an algorithm for morphing two polyhedral objects topologically equivalent to a sphere. First, two objects are mapped onto a sphere and merged by clipping one sphere to another. Then, a new shape is created that has combined graph information (vertices, edges, and faces) for the two objects. However, their technique applies only to star-shaped polyhedral objects. It is further extended to other types of objects such as tubular objects, which involve objects reconstructed from cylindrical scan-type range image. To establish correspondences over these types of objects, users must specify skeletons along the center of the contour of an object and project it to a convex-hull as an intermediate object. This approach can handle various types of objects, but it seems rather troublesome for users to specify such skeletons so as to establish correspondences, especially for geometrically complicated objects.

Parent [1992] presents a recursive algorithm that automatically finds a correspondence between the surfaces of two objects of equivalent topologies. This algorithm uses several sheets to cover the whole object. The two objects must have an equal number of sheets, and sheet boundaries must consist of the object's edges. Objects of genus g are automatically subdivided into 2(g+1) sheets, with each sheet recursively subdivided down to the face level. This approach completely establishes vertex-to-vertex interpolations and thus deformations between the two objects.

Delingette et al. [1993] use a simplex mesh. They propose basic mesh operations to alter shape topologies, thereby restricting their method to this type of mesh. The interpolation is performed using a physically based deformation approach to converting with a method derived from a data-fitting process. The process, however, seems time-consuming.

DeCarlo and Gallier [1996] describe another framework for the metamorphosis between two objects with different topologies, for example, the metamorphosis from a sphere to a torus. To make the surface correspondence between two objects with different topologies, each of two meshes is divided into a set of sub-meshes according to a set of the user-specified triangular or quadrilateral reference faces, called a control mesh. Metamorphosis between two objects with different topologies results from ensuring that each triangular face in one control mesh changes to the corresponding quadrilateral face in the other control mesh.

Lazarus and Verroust [1994; 1997] extend the work [Kent et al., 1992] for cylinder-like objects. Given a 3D curve inside each object, two cylindrical meshes are built to approximate the two objects. Their salient features being taken into account. The two

objects are morphed on these cylindrical meshes. The metamorphosis consists in an interpolation of two 3D curves composed by radial interpolation of each sampling point of the mesh.

Gregory et al. [1998] present an approach treating the metamorphosis between two objects with the same topological types. First users specify a network of curves, called a feature net, on each mesh. Each curve consists of vertices and edges of the mesh. The mesh is divided into several sub-meshes according to this net. Then area-preserving mapping establishes surface correspondences between each submesh of two objects. The user can generate desirable metamorphoses based on local refinements of a feature net.

Kanai et al. [1998] present a method with an automatic correspondence between two polyhedral meshes. Two 2-manifold objects are topologically equivalent to a sphere or disk. These objects are represented as a triangular mesh or a polyhedron with triangular faces. These two objects are assumed to have the same topology, called source object M_1 and target object M_2.

The approach to transforming from M_1 to M_2 usually involves dividing the problem into two steps. The first step is to establish a correspondence from each point of M_1 to a point of M_2. Once this correspondence is established, the next step involves a series of intermediate objects that are created by interpolating corresponding points from their original position to the target. The steps are the correspondence problem and the interpolation problem. The algorithm of [Kanai et al., 1998] for finding correspondence begins to develop M_1 and M_2 to a 2D unit disk D^2 and create embeddings by harmonic mapping. These 2D embeddings, called H_1 and H_2, have the same connectivity as M_1 and M_2, respectively, i.e., adjacent relations of vertices, edges, and faces are preserved. Next,

these two embeddings are merged, creating a new object H_c. It has inherited connectivity from both objects, and defines the point-to-point correspondence between each position of M_1 to that of M_2. H_c has not only the connectivity of $H_1(M_1)$, but also that of $H_2(M_2)$. By using H_c, the correspondence between H_1 and H_2 (and thus between M_1 and M_2) is established. By using H_c, continuous intermediate objects are created by interpolation. To establish an interpolation between M_1 and M_2, a simple linear interpolation technique is used [Kanai et al., 1998].

Future morphing work includes further work in morphing among multiple images, and improving automatic morphing among a limited class of images and video sequences. It would be useful to provide an interactive deformation tool with the capability of combining a sequence of deformations into a single one. It should also be possible to create a continuous warping between any two deformations. Such a tool could be used to establish a crude transformation between two shapes. Correspondence is an important stage for the animator. It tells which part of the source object will transform into which part of the target object. It should be possible to design an attractive tool to help the user greatly in the morphing process.

CHAPTER 6

THREE DIMENSIONAL SURFACE WARPING FOR PLASTIC SURGERY PLANNING

6.1 Introduction

6.1.1 Computer-Aided Planning for Plastic Surgery

Plastic surgery is a routine procedure to correct congenital deformities or to treat the deformities caused by accidents. The motivation of free-form deformation for precise pre-surgical planning is to help surgeons reduce the potential of risks in surgery and save the test time. For example, in nose and breast augmentation, it is highly desired to test different implants using 3D models and graphical user interface before the surgery. Currently, the patients have to tolerate the hurt and time-consuming test with various size and form implants during a surgery process. The implant material factory has to produce triple or four times of those that customs actually need. A surgeon also has to order up to a dozen of different implants for one patient in order to find the suitable one during the process. This work intends to provide one convenient method and computer tool to help simplify this process for surgeons.

Computer processing and analysis of medical imaging has been a hot topic for many years. Duncan et al. [2000] give one excellent review of the medical image analysis progress in the past two decades and point out the challenges ahead. In the image-guidance surgery area, there are several related studies. Peters et al. [1996] use multimodality imaging as an aid to the planning and guidance of neurosurgical procedures. Koch et al. [1996] propose algorithms in both presurgical planning and postsurgical evaluation. In their method, the facial surgery is realistically simulated using finite element methods. Sato et al. [1998] describe augmented reality visualization for

the guidance for breast-conservative cancer surgery using ultrasonic images acquired in the operating room just before surgical resection. St-Jean et al. [1998] develop one platform for preoperative and intraoperative use that permits manipulation of the merged atlas and MRI data sets in two and three dimensional views. Birkfellner et al. [2000] present a programming environment for computer-aided surgery and one application in oral implantology. Achermann et al. [1997] present one tool, which can be used to judge the success of nose surgeries. The tool is also used to investigate the long-term development of noses that undergo a plastic surgery. However, the tool is used only after the surgery and it handles two dimension images only. Lee et al. [1999] design several morphing operators. The augmentation operator is used on nose augmentation. They select nine features in source and target volumes. One operator is applied to control the effect of augmentation. However, it is very tedious, if not realistic, for surgeons to select that many feature points. To the authors' knowledge, there is no convenient tool for surgeons to perform 3D presurgical planning for plastic surgery. This work presents a research effort toward such a tool. It proposes to use direct manipulation of free-form deformation that was pioneered by [Hsu et al., 1992]. Free-Form Deformation (FFD) is first proposed by [Sederberg and Parry, 1986]. It supplies a general deformation frame. An object is embedded into an intermediate space, regarded as a deformation tool. To deform it, a user first deforms the embedding space; then the deformation is passed to the object. FFD technology is widely used in medical image analysis. For example, Rueckert et al. [1999] present an approach for the non-rigid registration of contrast-enhanced breast MRI. The global motion of the breast is modeled by affine transformation and the local motion is modeled by free-form deformation.

6.1.2 Surface Warping with Free-Form Deformation

As discussed in the previous chapter about image warping and morphing, deformation of an object's shape is one of the most important issues in geometric modeling. Deformations are powerful sculpting operations since they allow high-level shape modification, as opposed to manipulation of lower-level geometric entities. A particularly appealing type of deformation is the spatial deformation, which operates on the whole space regardless of the representation of the deformed objects embedded inside the space [Borrel and Rappoport, 1994].

Bechmann [1994] surveys different space deformation methods. Gomes et al. [1998] give an excellent survey on surface warping. They conclude with three different specifications: parametric specification, change of coordinates, and point specification. In the parametric specification, a global transformation of the euclidean space is defined by specifying the parameters for the transformation. The analytical equations of the transformation are obtained. The global transformations are typically taper, twist and bending. Since the parametric specification cannot easily achieve desired warping results, the following discussions focus on the other two specifications.

Change of coordinate: The warping of a graphical object can be interpreted as a change of coordinate systems. The coordinate change between curvilinear systems allows a rich variety of possibilities for warping. Also, the change of coordinate defines a global transformation of the space, meaning that the resulting warping can be applied to different classes of graphical objects embedded in the space, independent of their description or representation.

Barr [1984] first proposes the global and local deformation concept in which the transformation matrix is not constant but varies according to the space position. In other words, the deformation is represented as a matrix function of space position, independent of the geometric representation of the object. In the original FFD version [Sederberg and Parry, 1986], the intermediate space is a trivariate tensor product Bézier volume.

The process of warping a graphical object using change of coordinates can be attained in four steps:

(1) Define a new coordinate system on the space where the object is embedded;

(2) Map the object into the parametric volume space;

(3) Warp the representation curves of the coordinate system through editing control points in the lattice; and

(4) Compute the new object coordinates to Cartesian coordinates in order to reconstruct the deformed object.

In [Sederberg and Parry, 1986], a rectilinear system is represented by taking the coordinate curves uniformly spaced in each of the coordinate axes. Very often users need to deform some regions of the grid more than other regions. The work [Forsey and Bartels, 1988] solves the above problem by adaptively increasing the number of coordinate curves only in the regions that are subject to larger deformations. This technique starts with a coarse uniform rectilinear coordinate system that is incrementally refined in certain regions of the grid.

An extension to the work [Sederberg and Parry, 1986] is reported in [Coquillart, 1990]. Its basic idea consists in allowing lattices with different topology and geometry.

The use of more generic lattices allows the production of "shaped" warps to the surface, which is a very useful modeling tool.

A source and target lattices can be used for morphing. The lattice resulting from the morphing of the two lattices is used to deform the surface, which results in a morphing of the surface itself. The strategy for lattice morphing is exploited in [Coquillart and Jancene, 1991].

Another work that enables free-form deformations between lattices of arbitrary topology appears in [MacCracken and Joy, 1996]. The proposed technique uses an extension of the Catmull-Clark subdivision methodology that successively refines a 3D lattice into a sequence of lattices that converge uniformly to a region of 3D space. Deformation of the lattice then implicitly defines a deformation of the space. An underlying model can be deformed by establishing positions of the points of the model within the converging sequence of lattices and tracking the new positions of these points within the deformed sequence of lattices.

Point specification: The warping techniques based on free-form deformation allow localized transformation using B-splines or Bézier coordinate systems. The deformations are defined by parametric functions (3D splines) whose values are determined by the location of control points. Describing a free-form deformation in conventional modeling systems is done by manipulating these control points. The interface that reflects the underlying mathematics of the modeling method can be confusing because the control point movement merely hints at the type of deformation to which the object is subjected [Hsu et al., 1992]. In order to deform the coordinates, a user needs to anticipate how the surface should be deformed. Although the movement of the control points gives an

indication of the resulting deformation, some shapes are not intuitive to form. This can be exemplified in the case where it is needed to deform a surface to produce a flat bump. The user probably would position the control points aligned with the plane, as shown in Figure 6.1(a). However, to obtain the desired flat bump deformation, the control points should be positioned as shown in Figure 6.1(b).

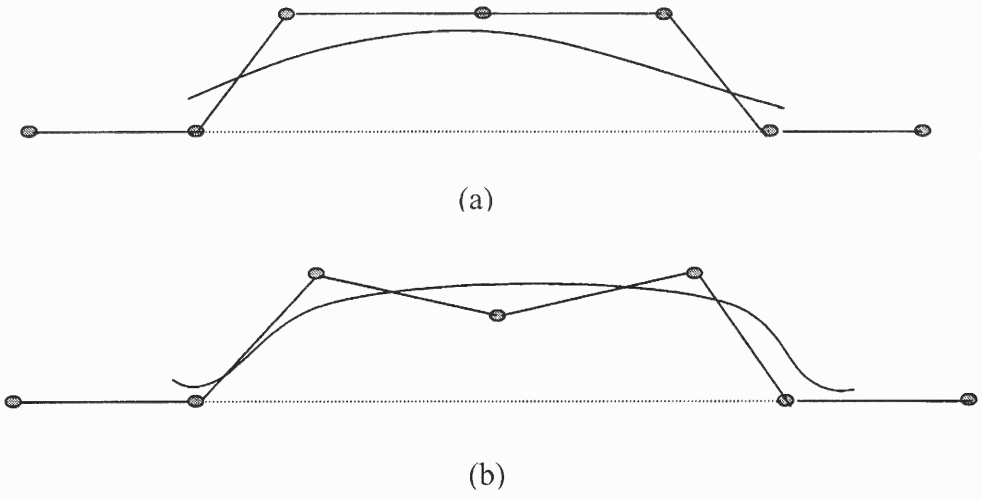


Figure 6.1 Two different control point configurations. The dashed line shows the original shape. (a) One flat line control point deformation result; (b) The control point configuration to create a flat top [Hsu et al., 1992].

Thus there are four problems in manipulating deformations via control points [Hsu et al., 1992]:

(1) Exact shape is difficult to achieve;

(2) Exact placement of object points is difficult to achieve;

(3) Users unfamiliar with splines do not understand the purpose of the control points and the results of their movement; and

(4) The control points become difficult to manipulate when occluded by the object being deformed, or when there are so many they clutter the screen.

Hsu et al. [1992] present their pioneer work in Direct Manipulation of Free-Form Deformation (DFFD). Warping by direct manipulation converts a user action of the form "move this object point to there" and finds the lattice points change to meet the action result. The deformed location of the selected point matches the target point location. It is important that users do not interact with the coordinate system, but only with the selected object point at which the transformation is specified.

This work proposes for the first time a DFFD based method for 3D surface warping in plastic surgery planning.

6.2 A DFFD Based 3D Surface Warping Method

Motivated by the Direct Manipulation of Free-form Deformation method [Hsu et. al., 1992], this work proposes a DFFD based 3D surface warping method for pre-surgical planning. It is mainly composed of the following five parts:

(1) B-spline used to define the model surface;

(2) Warping by direct manipulation method;

(3) Scalable lattice resolution to control deformation area;

(4) Displacement vector that can flexibly change to achieve different results; and

(5) Volume calculation in the process of warping;

Bézier and B-spline are two different free-form coordinate systems. They are discussed below.

6.2.1 Bézier Free-Form Coordinate System

Consider a rectangular region R of the plane and define on R a representation of a Cartesian coordinate system using a lattice of (m+1)(n+1) points P_{ij}, $0 \le i \le m$, $0 \le j \le n$.

A point p of the representation lattice has coordinates (i/m, j/n), with i=0,…,m. and j=0,…,n. A free-form coordinate system on the region R by using points P_{ij} as control points of Bézier curves.

The Bézier curve b(t) determined by n+1 control points P_0,…, and P_n is defined by

$$b(t) = \sum_{i=0}^{n} p_i B_i^n(t) \tag{6.1}$$

where $B_i^n(t)$ is the Bernstein polynomial of degree n:

$$B_i^n(t) = \frac{n!}{i!(n-i)!} t^i (1-t)^{n-i}, i = 0,...,n. \tag{6.2}$$

Extension of the Bézier reconstruction equation to higher dimensions is attained using a tensor product. In this way, the free-form coordinates (s, t) introduced on region R are related with the Cartesian coordinates by

$$p = (x,y) = W(s,t) = \sum_{i=0}^{m} \sum_{j=0}^{n} P_{ij} B_i^m(s) B_j^n(t). \tag{6.3}$$

Generalization of the above equation to third dimension is immediate:

$$p = (x,y,z) = W(s,t,u) = \sum_{i=0}^{m} \sum_{j=0}^{n} \sum_{k=0}^{q} P_{ijk} B_i^m(s) B_j^n(t) B_k^q(u). \tag{6.4}$$

Initially, the coordinate curves of the Bézier and the Cartesian coordinate systems coincide, but the Bézier coordinates are normalized to the interval [0,1]. The warping is specified by moving the lattice points P_{ijk}, so as to obtain new points P_{ijk}'. These new points substitute for the old ones in Equation 6.4, which is used to compute the new deformed coordinates.

It should be noted that the use of Bernstein polynomials makes the warping transformation global. As a consequence, the modification of a single control point influences the whole domain. In plastic surgery application, this property of Bézier splines becomes a drawback, making it impossible to control the surgery area during simulation. Since B-spline can overcome this drawback, it becomes the unique choice to use in the proposed method. Next the detailed B-spline free-form coordinate system is discussed.

6.2.2 B-spline Free-Form Coordinate System

A Bézier free-form coordinate system does not have good localization properties and the degree of the reconstruction basis (Bernstein basis) increases with the number of points in the representation lattice. B-spline solves these two problems and is thus used to model free-form surfaces.

Let Ω be a domain of the image volume in the *uvw*-plane that contains points p=(u,v,w) such that $1 \le u \le m$, $1 \le v \le n$, and $1 \le w \le q$. Let Φ be a $(m+2)\times(n+2)\times(q+2)$ lattice of control points overlaid on the region Ω. In the initial configuration of Φ, the *ijk*-th control point lies at its initial position (i, j, k) in the *uvw*-plane. Then, the FFD can be written as the 3-D tensor product of the familiar 1-D cubic B-splines:

$$P = (x, y, z) = \sum_{a=0}^{3} \sum_{b=0}^{3} \sum_{c=0}^{3} B_a(r) B_b(s) B_c(t) \Phi_{(i+a)(j+b)(k+c)} \tag{6.5}$$

where i= [u]-1, j = [v]-1, k = [w]-1, r = u-[u], s = v-[v], and t = w-[w], and B_a represents the *a*th basis function of the B-spline as follows:

$$
\begin{aligned}
B_0(u) &= u^3/6 \\
B_1(u) &= (1+3u+3u^2-3u^3)/6 \\
B_2(u) &= (4-6u^2+3u^3)/6 \\
B_3(u) &= (1-u)^3/6.
\end{aligned}
\tag{6.6}
$$

If the coordinate curves are described by a B-spline function, it is necessary to add control points to the grid, because the B-spline curves do not interpolate the first and last control points of the coordinate curves. It is possible to add these control points using the technique of ghost vertices. The coordinates of these vertices are computed such that the B-spline curve interpolates the first and last control points.

6.2.3 Warping Using Direct Manipulation

An object to be warped needs to be embedded in one coordinate system. The coordinate system is represented with lattice points. It is necessary to find the displacement vector for each control point of the lattice that would produce the desired deformation. If the coordinate curves of the lattice are described by B-spline, then the deformed position q of any arbitrary point with local coordinates, (s,t,u), is given by:

$$
q_{i,j,k}(s,t,u) = \sum_{l,m,n=-3}^{0} P_{i+l,j+m,k+n} B_l(s) B_m(t) B_n(u) \tag{6.7}
$$

where $P_{i,j,k}$ is the i^{th}, j^{th}, k^{th} control point in the s, t, and u direction, respectively, and the B_l, B_m, and B_n are the B-spline blending functions.

Certainly there are many control-point configurations that yield the same deformed location for the selected point. A natural coordinate change is the one that moves the lattice control points the least in the sense of least squares. The blending function of

Equation (6.7) assigns weights to the control points for a given target point. The closer the control point is to the target point, the greater the influence of that control point.

From Equation (6.7), the deformed object point location q is a linear function of 64 control points P. q can be written in the matrix form of q = BP, where B is a 1×64 matrix that contains the product of the spline functions and P is a matrix 64×3 that contains the coordinate of the control points. Given a point Q = (x, y, z) of the object surface, and its new location Q' = (x', y', z'), determined by a displacement vector, then Q' is given by q' = B (P + ΔP), or

$$\Delta q = B\Delta P \tag{6.8}$$

where ΔP is the change in position of the control points and Δq is the change in position of the object point, i.e., the difference between points q and q'. To solve equation (6.8) and find the value of ΔP, it is practical to use a pseudoinverse (or generalized inverse) B^+ of B. Then the value of ΔP can be solved by

$$\Delta P = B^+\Delta q \tag{6.9}$$

Given a system of linear equations y = Bx, the pseudoinverse B^+ is a matrix where $x_0 = B^+y$ is the best solution, in the least squares sense, to the system of equations. Under this solution, $\|Bx_0 - y\|$ is minimized and $\|x_0\|$ is as small as possible. The pseudoinverse of the single-row matrix B can now be found by the equation

$$B^+ = \frac{1}{\|B\|^2} B^T \tag{6.10}$$

Because the pseudoinverse gives a least-squares solution, the change in control point positions is minimized.

In the proposed method, one displacement vector can be divided into several warping steps. The lattice resolution adjusts the surface area. Those two features make the proposed method more flexible in pre-surgical planning than the original DFFD. The following discussions focus on these two features.

6.2.4 Lattice Resolution

In the implementation of DFFD, users define one lattice. Lattice resolution can directly affect the deformation area. In this work, the lattice space in x, y, and z directions are different. In Figure 6.2, the resolution is 3×3×3. The whole face is deformed. While in Figure 6.3, the lattice resolution is 12×12×12, small area around nose is deformed. The proposed method can change the lattice resolution in different steps if necessary by dragging the slider in graphical user interface to be discussed later.

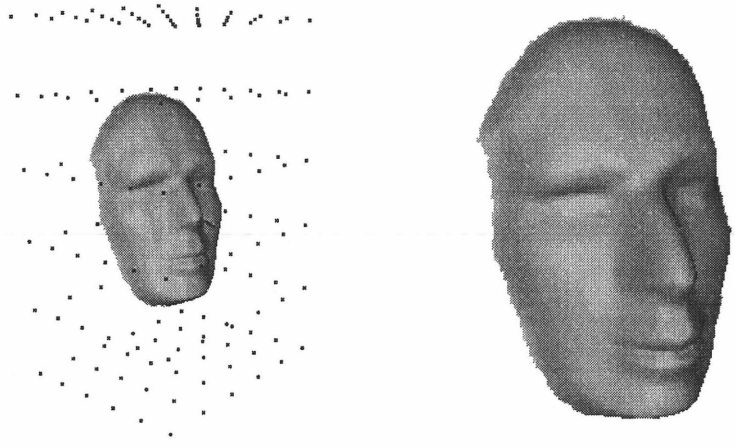


Figure 6.2 3 × 3 × 3 lattice points and the deformation result that affect the entire face.

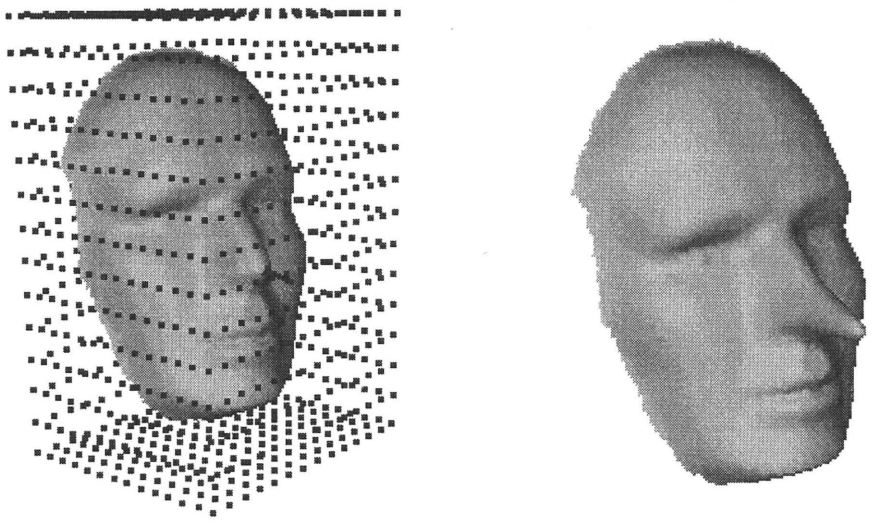


Figure 6.3 12 × 12 × 12 lattice and the deformation result that affect locally.

6.2.5 Displacement Vector

In this application, the length and direction of a displacement vector can be changed randomly. An entire deformation process can be separated into several sub-deformation processes. Each sub-process deforms some percentage of the overall displacement vector by dragging the slider. During the sub-processes, the user still can alter the displacement vector and lattice resolution according to the result. The combination of different vector direction and length, the lattice resolution offers great flexibility. Figure 6.4 shows one displacement vector on the nose tip.

The entertainment industry can also use this free-form deformation application. For example, in movie, game or cartoon, computer graphics engineers obtain various eccentric faces by deforming different part on the face. Additionally the displacement vector helps much to deform the specified part. In Figure 6.5, the funny-looking nose is

obtained by deforming the overall displacement vector in several steps. The effect of deformation in Figure 6.5 is achieved by following steps as in Table 6.1. The lattice resolution can also be changed in every step.

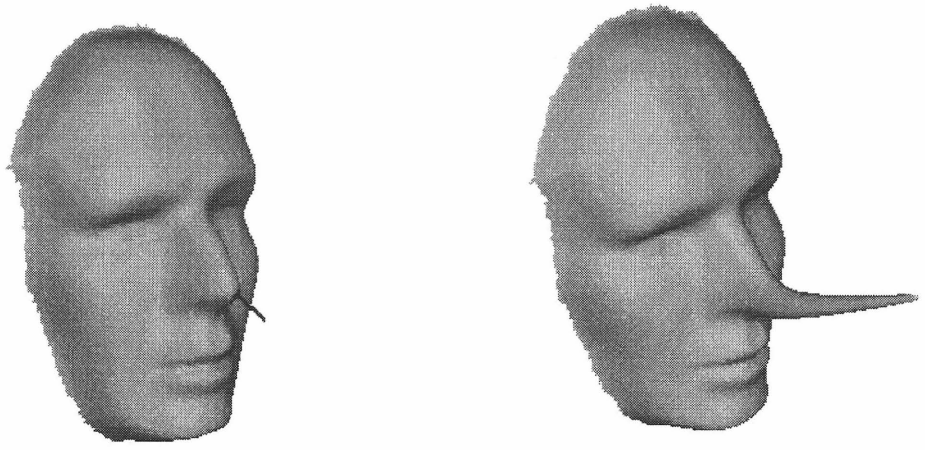


Figure 6.4 Displacement vector. **Figure 6.5** One multi-step deformation example.

Table 6.1 Five step deformation parameters.

No.	Vector length (Percentage)	Lattice resolution
1	10	5x5x5
2	20	7x7x7
3	20	9x9x9
4	30	11x11x11
5	20	14x14x14

The closest work to such effect is documented in [Lee et al., 1999]. The drawback of their method is the difficulty for users to specify nine feature points in source and target volumes.

6.2.6 Volume Change

The volume change can be obtained approximately. For one triangle on the facial mesh in one small deformation step, the volume change of facial model is roughly specified in Figure 6.6. Where B is the triangle on the model surface before deformation. A is the triangle after the triangle B is deformed. n_1 and n_2 are the normals of A and B.

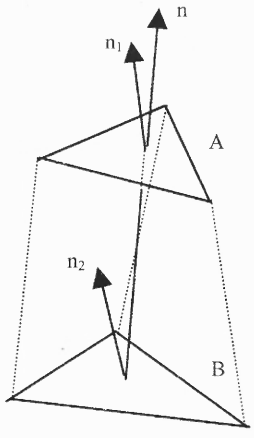


Figure 6.6 The volume change approximation.

$$\Delta V = 0.5\ (S_{A1}+S_{B1})\ L \tag{6.11}$$

$$S_{A1} = S_A\ (n \bullet n_1) \text{ and } S_{B1} = S_B\ (n \bullet n_2) \tag{6.12}$$

Where L is the vector length between the centers of triangle A and B. S_A and S_B is the area of triangle A and B, respectively.

In Figure 6.7, the volume change caused by one triangle and the whole volume change in the warping process are specified respectively.

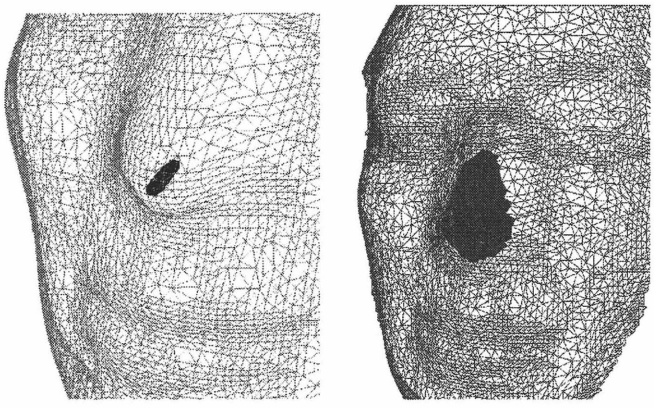


Figure 6.7 The volume change of one triangle and whole deformation.

6.3 Nose Augmentation Application

Rhinoplasty, or surgery to reshape the nose, is one of the most common of all plastic surgery procedures. Rhinoplasty can reduce or increase the size of nose, change the shape of the tip or the bridge, narrow the span of the nostrils, or change the angle between the nose and the upper lip. It may also correct a birth defect or injury, or help relieve some breathing problems.

The goal of the free-form deformation technique in nose augmentation surgery is to give the surgeons and patients one planning tool before the surgery. After obtaining the three dimensional face image of the patient, the surgeons can modify the nose size and shape according to the requirement. Due to the technological advances, 3D facial data becomes less and less expensive to obtain. This work uses the camera of Genex Technologies Inc. to obtain the 3D facial data, as discussed next.

6.3.1 3D Facial Data

The 3D patient facial data can be scanned using Rainbow 3D camera. If original 3D data points are too dense, it is necessary to compress the 3D mesh vertices while keeping the model almost same as the original one.

Figure 6.8 3D image system.

The Rainbow 3D Camera (Figure 6.8) developed by Genex Technologies, Inc.(GTI), is a novel three-dimensional surface profile measurement system capable of acquiring full frame dynamic 3D images of objects with complex surface geometry at a high speed. A "Full Frame 3D image" means that the value of each pixel (i.e., picture element) in an acquired digital image represents the accurate distance from the camera's

focal point to the corresponding point on the object's surface. The (x, y, z) coordinates for all visible points on the object surface are provided by a single 3D image.

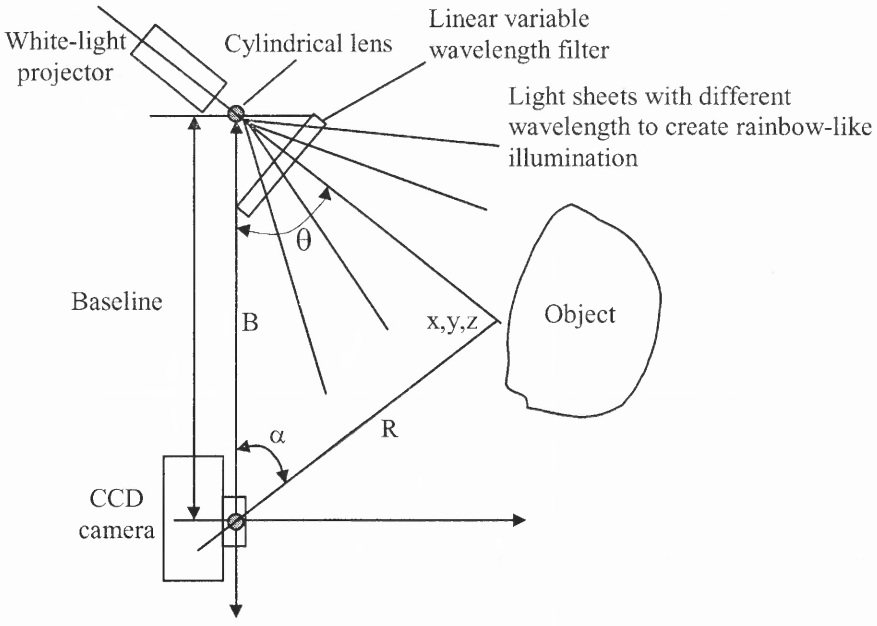
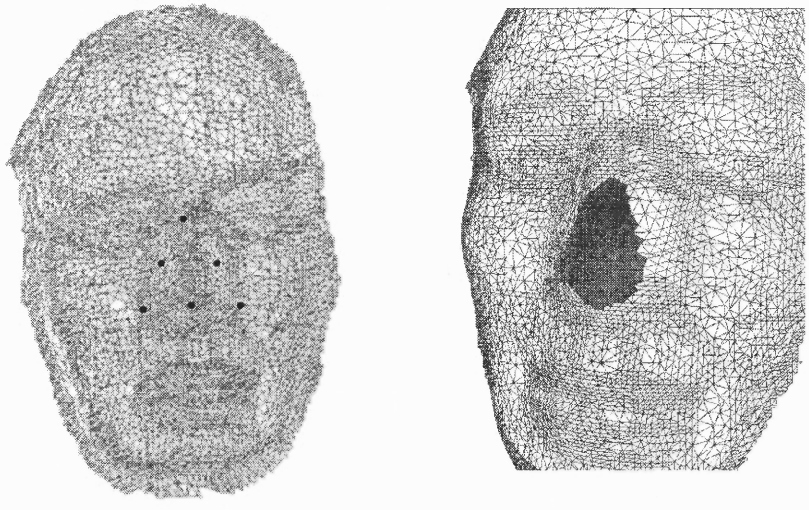


Figure 6.9 Rainbow imaging system.

In Figure 6.9, the light from the projector is sent through a cylindrical lens arrangement followed by a linear variable-wavelength filter. The color of the projecting light is a spatially and continuously varying wavelength, and color is encoded with a particular light-projection angle. Wavelength information is then encoded with angle information based on a triangulation principle. The formed light reflected from the object is then digitized and used to extract 3D information. It costs several seconds to scan one 3D facial data.

The 3D face data consists of vertex coordinates and three vertex indexes of every triangle. There is no information about the vertex location in the data source. It is very

difficult to detect the nose vertices in the whole data set. In Figure 6.10(a), the critical points to define the nose form are specified manually. It is one very tough work to detect them in 3D original data. There may be some methods to detect them roughly. However, the deformation result is not ideal even only small error exists using such automatic extraction methods. How to confine the nose deformation area is a critical issue in the whole application. Lattice points of different resolution can be used on the same object to create different curvatures and deformation area.



(a) (b)

Figure 6.10 The control of deformation area.
(a) The 3D wireframe facial data with critical points on nose.
(b) One example of deformation area control by adjusting lattice resolution.

In Figure 6.10(b) the volume change area is specified on the facial model. The lattice resolution is 8×8×8. This result is ideal in the control of nose deformation area.

This work uses a 3D facial model that consists of 6305 vertices and 12202 triangles.

6.3.2 Nose Augmentation Method

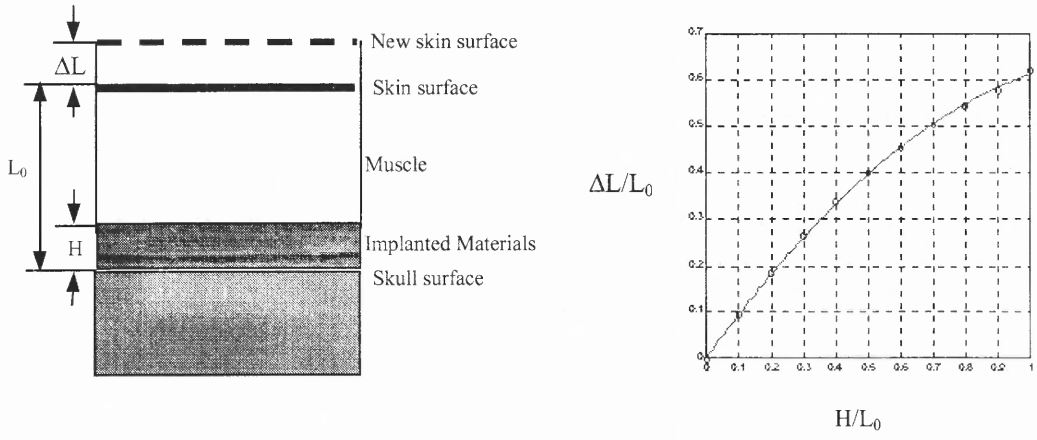
The vertices on nose are very difficult to be detected. It is a feasible method to control the deformation area by changing the lattice point resolution. In the user interface, the surgeons select one point on nose surface. One vector is generated on this point. The surgeons can change the vector direction and length by dragging the slider. During the deformation process, users can still change the direction in order to adjust the nose shape. One whole deformation process can be separated into several steps according to the need during this process. The lattice point resolution can also be adjusted in any time during the deformation.

6.3.3 Simplified Skin-muscle Model

After the warping result is achieved using DFFD based warping method, it is necessary to provide information on the different tissue types and compute the rough implanted material form and size. Lee et al. [1995] propose an elaborate 5-layed soft tissue model with biphase springs. The tissue model is composed of epidermal surface, dermal-fatty layer, fascia surface, muscle layer and skull surface. Koch et al. [1996] propose a 3-layer spring mesh model.

In order to improve the computation efficiency, this work proposes a simplified skin and muscle model. The skull is covered by deformable tissue that consists of muscle and skin. In the nose augmentation surgery, bone is implanted to enhance some part of the nose. The bone is supposed to be implanted right on the nose skull. Koch et al. [1995] present the spring stiffness parameters of different tissue type. Skin stiffness is 200.0, muscle stiffness is 100.0, and bone stiffness is infinite. According to these parameters and

the physical property analysis, the muscle displacement thickness caused by bone implantation is less than the implanted bone thickness. The specified 2D facial model is shown in Figure 6.11(a), the skin surface is deformed to new skin surface after bone implantation. *H* is the implanted bone thickness, ΔL is the displacement thickness of skin and muscle together. L_0 is the thickness between skull surface and skin surface. The larger is H/L_0, the smaller is $\Delta L/L_0$.



(a) (b)

Figure 6.11 The simplified skin-muscle model.
(a) The simplified facial model in 2D;
(b) The relationship between H/L_0 and $\Delta L/L_0$.

Approximate relationship between H/L_0 and $\Delta L/L_0$ is shown in Figure 6.11(b), using curve fitting method, the equation between these two parameters is as follows:

$$\Delta L/L_0 = -0.36503\ (H/L_0)^2 + 0.9741\ (H/L_0) + 0.0025$$

Since the exact skull data set is not available in the present work, the skull surface is simulated by projecting every original triangle onto one smaller triangle along the inverse normal direction. Both skull structure and more accurate facial model are the future topics.

6.4 Implementation and Illustration

The software application is developed in C++/MFC and OpenGL on windows platform. A personal computer with a Pentium II 450MHz CPU and 128M memory is used.

6.4.1 User Interface

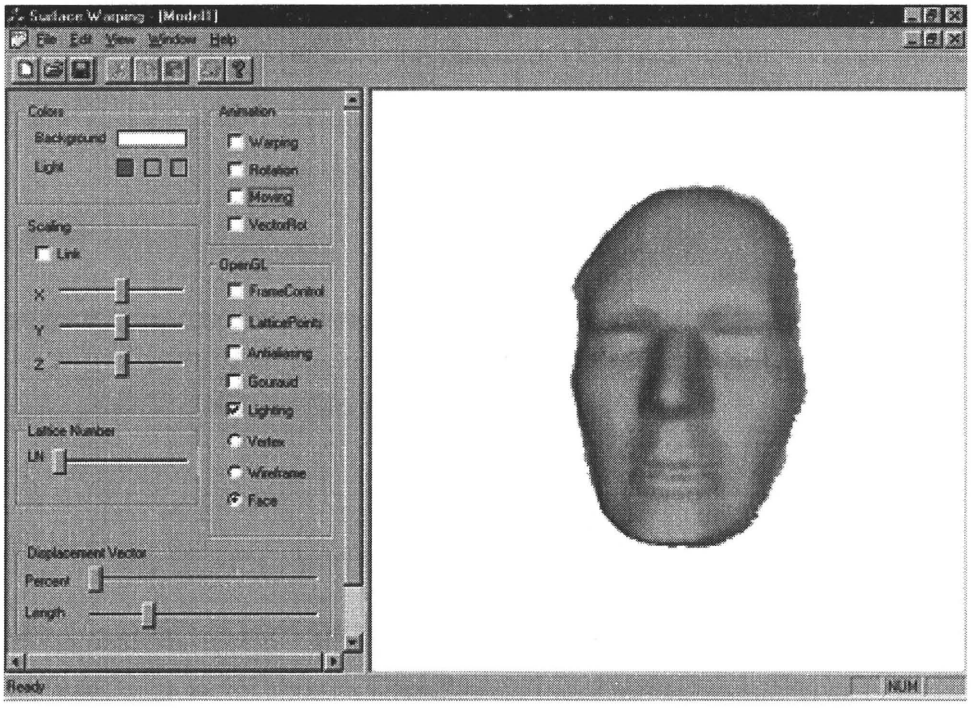


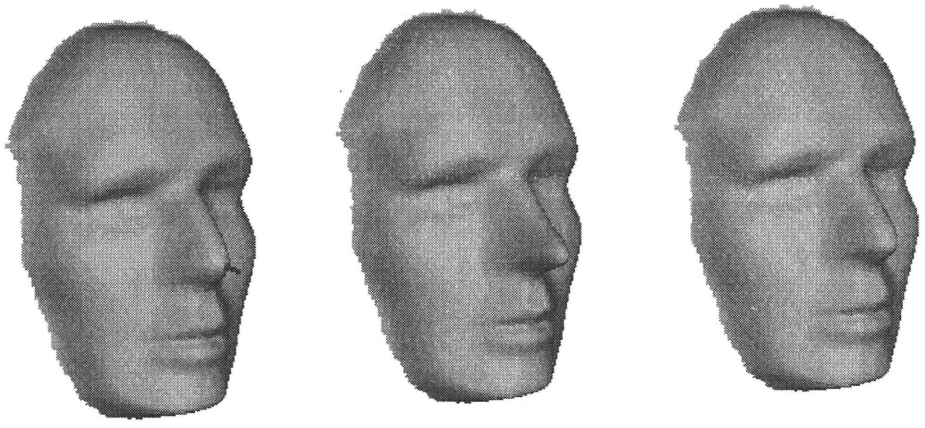
Figure 6.12 The user interface of nose augmentation application.

The object model can be viewed in wireframe, vertex and face format. Users can rotate, transform and scale the object and select the background and light color. As shown in Figure 6.4, an object point is selected and a vector is attached on this point. By dragging the "Length" slider and clicking the "VectorRot" box, the length and direction of displacement vector can be changed as desired. One whole deformation process can

consist of several deformation sub-processes. Each sub-process deforms some percentage of the overall deformation by dragging the "Percent" slider.

6.4.2 Illustrative Examples

In Figure 6.13(a), one small displacement vector is applied to the nose's selected point. The lattice resolution is 12×12×12. The final nose is augmented in Figure 6.13(b). The augmented nose can also be reduced to Figure 6.13(c) by changing the displacement vector to inverse direction.



(a) (b) (c)

Figure 6.13 Nose augmentation example.

Compared with other documented methods as in Table 6.2, the proposed method has the advantages of the previous methods. It is computationally fast suitable for real-time pre-surgery applications.

Table 6.2 Different method comparison.

Method	Dimension	Pre/post surgery	Control points selection	Deformation area control	Notes
[Achermann et al., 1997]	2D	Post-surgery	N/A	N/A	Only for post-surgery analysis
[Lee et al., 1999]	3D	Pre-surgery	Difficult	Yes	Difficult to select feature points
[Hsu et al., 1992]	3D	N/A	Easy	Yes	Have not applied to plastic surgery
This method	3D	Pre-surgery	Easy	Yes	Flexible in pre-surgical planning with real-time speed

The skin-muscle model is discussed in the above section. Figure 6.14 presents one example of simulated implanted bone form and size. The simulation is in real-time speed (milliseconds). Figure 6.14(a) shows the original mesh. Figure 6.14(b) presents the nose deformation and the implanted material based on the proposed skin-muscle model. Figure 6.14(c) and (d) are the rendered facial images before and after the deformation.

6.5 Volume Preserving Surface Warping

The free form deformation method used in work needs to consider the volume preservation problem. In the application of plastic surgery, for example, breast enhancement, it is a very important step to give the accurate deformation result. The volume of a water bag implanted into breast should not be changed during the

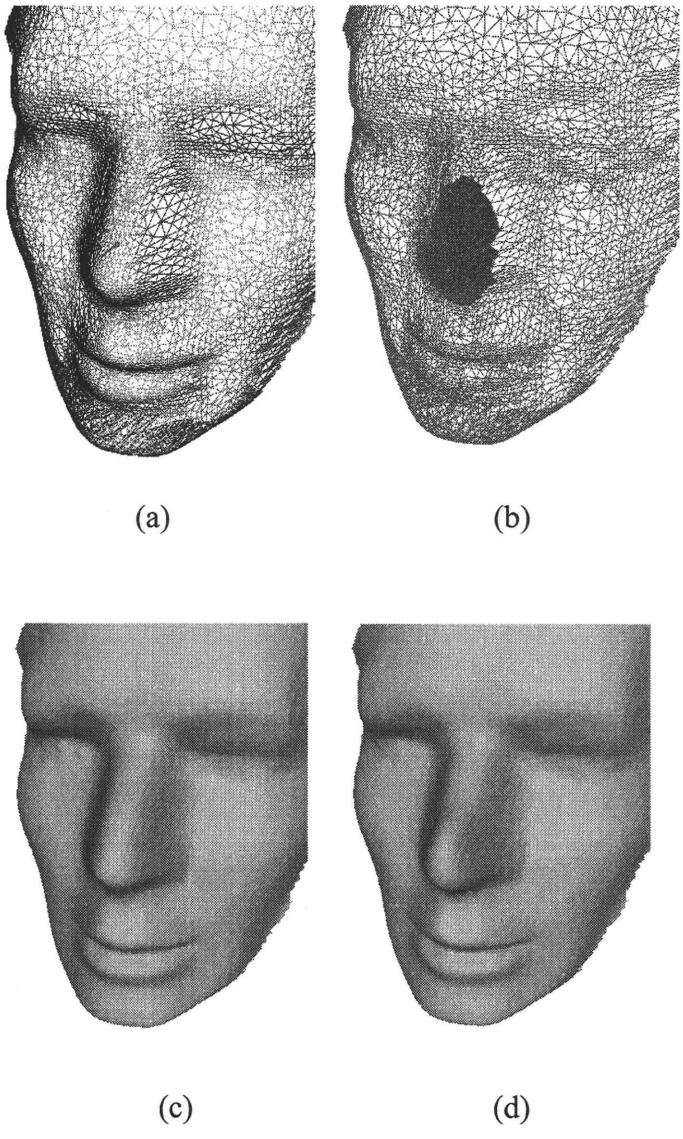


Figure 6.14 One example of implanted material simulation.
(a) the mesh before deformation; (b) the implanted material specified;
(c) and (d) The rendered facial model before and after the deformation.

deformation process. There are few works done to preserve the volume of objects throughout the deformation. Sederberg and Parry [1986] propose the FFD method. The model in their work allows one to compute easily the volume variation, but the definition of deformations that actually preserve the volume is not yet possible. The authors prove

that the volume is preserved when the Jacobian of the deformation function is equal to 1. However, it is not explained how to obtain a unit Jacobian while the user moves control points of the deformation tool and it does not seem that easy.

Rappoport et al. [1996] propose one volume preserving method. The volume preservation is achieved for objects defined with Bezier solids. They present a method for modeling an object composed of several tensor-product solids while preserving the desired volume of each primitive and ensuring high-order continuity constraints between the primitives. But the problem is tackled only for a solid modeling tool, not an independent deformation function. Therefore, this approach is not ideal for the easy use.

Aubert and Bechmann [1997] propose one volume-preserving space deformation method. Their method is based on the space deformation model called DOGME. A feature of DOGME is used to solve the problem of volume preservation. DOGME is based on a direct manipulation of the space points by imposing linear constraints. These constraints are solved by using pseudo-inverse. Although this model is totally independent of object representation, it allows people to integrate some characteristics of objects. The optimization term in the mathematical expression of the deformation is directly used to solve the problem of volume preservation. Their algorithm is introduced in more detail as follows.

6.5.1 The Space Deformation Model DOGME

With the model DOGME, a deformation is defined in an *n*-dimensional space by a displacement function d(U) that is associated with each point U of R^n. The deformation transforms a point U of R^n into a point U' = U + d(U). R^3 is considered in this work. The

function f transforms the points from R^3 to R^m, for any dimension m. f is called an extrusion function. The transformation T transforms the points from R^m into R^3. M represents the matrix of T and is called a projection matrix. d can be expressed in a matrix product:

$$d(U)_{(3\times1)} = M_{(3\times m)}\, f(U)_{(m\times1)} \tag{6.13}$$

f(U) is calculated by a tensor product of B-splines. Suppose p_k B-splines of degree d_k are uniformly distributed along each coordinate u_k ($k\in[1,3]$ in 3-D space). $f^k(u_k) = (B_0^{d_k},...,B_{p_k-1}^{d_k})'$ with $k\in[1,3]$ where ' represents the matrix transpose. Then f is defined by: $f(u) = \otimes_{k=1}^{3} f^k(u_k)$. Or, with a more explicit expression:

$$f(u) = \begin{pmatrix} B_0^{d_0}(u_0)B_0^{d_1}(u_1)B_0^{d_2}(u_2) \\ B_1^{d_0}(u_0)B_0^{d_1}(u_1)B_0^{d_2}(u_2) \\ B_0^{d_0}(u_0)B_1^{d_1}(u_1)B_0^{d_2}(u_2) \\ \cdots \\ B_q^{d_0}(u_0)B_r^{d_1}(u_1)B_s^{d_2}(u_2) \\ \cdots \\ B_{p_0}^{d_0}(u_0)B_{p_1}^{d_1}(u_1)B_{p_2}^{d_2}(u_2) \end{pmatrix}$$

In 3D cases, B-splines of degree 3 are used. If seven B-splines are set along each of X, Y and Z axis, that involves a dimension of m=343 for the extrusion function f. Once f is defined, the matrix M is calculated so as to achieve some given point displacements. These displacements, called displacement constraints, are fixed for given points, called constraints points. The displacement constraint is a spatial displacement and independent of the object to be deformed. So the displacement constraint is not attached to a point of the object. This kind model provides an intuitive tool to deform an object in response to *compress* or *expand* operations.

6.5.2 The Optimization Term

The matrix M is computed in order to satisfy the displacement constraints. $(V_i)_{i\in[1,nc]}$ represents n_c constraint points and $d(V_i)$ the displacement vector imposed on point V_i. According to the definition of the deformation function, the n_c constraint points V_i must satisfy equation (6.13) as follows:

$$d(V_i) = M\, f(V_i) \quad i\in[1, n_c] \tag{6.14}$$

Transposing each of these n_c equations gives:

$$d'(V_i) = f'(V_i)M' \quad i\in[1, n_c] \tag{6.15}$$

Let

$$X = \begin{pmatrix} f'(V_1) \\ \vdots \\ f'(V_{n_c}) \end{pmatrix} \text{ and } D = \begin{pmatrix} d'(V_1) \\ \vdots \\ d'(V_{n_c}) \end{pmatrix} \tag{6.16}$$

The following matrix equation is obtained:

$$D_{(n_c\times 3)} = X_{(n_c\times m)} M'_{(m\times 3)} \tag{6.17}$$

In equation (6.17) $M'_{(m\times 3)}$ is the unknown. The pseudo-inverse X^+ of X is used to solve the equation.

$$M'_{(m\times 3)} = X^+_{(m\times n_c)} D_{(n_c\times 3)} + (I - X^+X)_{(m\times n)}\, \xi_{(m\times 3)} \tag{6.18}$$

where I is the identity matrix and ξ is an arbitrary matrix in $R^{m\times 3}$. The term $(I - X^+X)\xi$ defines the optimization term.

Equation (6.17) admits an infinity of solutions, the optimization term allows one to describe the set of solutions with the parameter ξ. For any value of ξ the displacement constraints are always achieved. In a general case, the choice of the tensor product of B-splines for the extrusion function f assures an infinity of solutions. This extrusion

function may lead to a non-infinity set of solutions. With DOGME, when it is not necessary to take advantage of the optimization term, ξ is simply chosen to be zero. But by taking an appropriate value for ξ, one is able to impose some constraints to the deformation in addition to the displacement constraints. A formulation of ξ that achieves the preservation of the volume of objects is used.

The expression of the displacement of a point in R^3 deformed by DOGME, according to the expression (6.14) and (6.18) is as follows:

$$d'(u) = f'(u)(X^+D + (I - X^+X)\xi)$$

In the following discussion, the expression of the j^{th} coordinate of the displacement d(u) is used more precisely. To simplify the notation, dU_j is written instead of $d(U)_j$:

$$dU_j = f'(U)(X^+D_j + (I - X^+X)\xi_j) \quad j \in [1,3]$$

Consequently, objects deformed by this function always verify the displacement constraints. The problem is then to simultaneously add the volume preserving constraint. In the following section, the object representation and volume variation between initial objects and deformed objects are introduced.

6.5.3 Representation of Objects and Calculation of the Volume Variation

In the volume-preserving algorithm, the objects deformed by DOGME are represented by their boundaries (B-rep). This boundary has to be a closed and orientable surface in order to define properly the volume of an object. To keep the intuitive notion of volume, the surface is supposed not to intersect itself.

The deformed object P' is obtained from a given object P by applying the deformation function d only to the vertices of the polyhedron P. The facets of P' remain

triangular and plane. Finally, a facet of P' is simply given by a facet of P such that its three vertices are moved. The volume of the object P is preserved throughout a deformation if the volume of the polyhedron P is equal to the volume of the polyhedron P'. The calculation of the volume is given below.

Let f be the set of facets of the object P. If f is an element of F, then f is defined by its three vertices (A_f, B_f, C_f). The orientation on the object P implies an orientation on the facet f. If f is oriented in the direction A_f, B_f, C_f, then the directed normal $n_f = A_fB_f \wedge B_fC_f$, where $\wedge$ denotes the cross product. The volume of P is given by:

$$V(P) = \frac{1}{6} \sum_{f \in F} A_f \cdot n_f$$

The difference between V(P) and V(P') provides the volume variation between P and P'. One more suitable calculation can be chosen. The formulation of the volume variation can be obtained directly by simply noticing that P and P' are defined by a unique polyhedron whose vertices have been displaced.

The displacement of one vertex that belongs to the polyhedron P involves a volume variation. The deformed object P' is obtained by moving successively each vertex of P . If the volume variation is caused in each displacement, the volume difference between P and P' is obtained.

Each facet displacement contributes to the volume variation. Let (A,B,C) be a facet of P. The displacement order imposed to the vertices of P implies a displacement order onto the vertices {A, B, C}. Suppose that A is displaced towards A', before B is displaced towards B', and that finally C is displaced towards C'. The triangle (A', B', C') is a facet of the deformed object P'. The orientation on the object P also induces an orientation on (A, B, C). Suppose that this orientation is A, B and C. If the facet (A, B, C) is considered

only, then the displacement of A into A' produces a volume variation equal to the tetrahedral volume (A, B, C, A'). This volume is given by:

$$V(A,B,C,A') = \frac{1}{6}(CA \wedge AB) \cdot AA'$$

This volume is algebraic and the cross product must be calculated according to the orientation chosen for the facet. The calculation carries on by displacing B towards B'. This displacement involves a volume variation equal to V(A', B, C, B') because A is already displaced towards A. In the same way, the displacement of C towards C' causes a variation equal to V(A', B', C, C'). The contribution to volume variation of the facet (A, B, C) is the sum of these three tetrahedral volumes. If B is displaced before A, neither the same tetrahedral decomposition nor the same value for the sum of the three tetrahedral volumes is obtained. In order to get the right volume variation of the whole object P, it is necessary to follow for each facet contribution the displacement order imposed on the vertices of P. Nevertheless the global volume variation of the whole object P is always the same for all displacement orders.

To express more simply the volume variation, a displacement vector is used. Thus, from the example of the facet (A,B,C), the displacement vector AA' is a factor of the cross product $CA \wedge AB$. Suppose that α_A is the sum of the cross products that are related to AA', and dA is the displacement vector AA'. Then, the volume variation of P is expressed as a sum of terms like $\alpha_A \cdot dA$ on all vertices of the object P.

Let $(U_i)_{i\in[1,n]}$ be the n vertices of the object P and dU_i the displacement vectors of the points U_i. α_i is the sum of the cross products that are related to dU_i. Finally, the expression $\Delta V(P)$ of the volume variation of the object P as a function of the vertex displacements:

$$\Delta V(P) = \frac{1}{6}\sum_{i=1}^{n} \alpha_i \cdot dU_i \tag{6.19}$$

Since there exist six ways to arrange the three vertices of a triangle, the facet contributions can be computed in six different ways. To remove the displacement order, the new facet contribution is defined by the sum of the six possible calculations. In the example of the facet (A,B,C) displaced to (A',B',C') the six possible orders are:

- displacement order A,B,C: V(A,B,C,A') + V(A',B,C,B') + V(A',B',C,C')
- displacement order A,C,B: V(A,B,C,A') + V(A',B,C,C') + V(A',B,C',B')
- displacement order B,A,C: V(A,B,C,B') + V(A,B',C,A') + V(A',B',C,C')
- displacement order B,C,A: V(A,B,C,B') + V(A,B',C,C') + V(A,B',C',A')
- displacement order C,A,B: V(A,B,C,C') + V(A,B,C',A') + V(A',B,C',B')
- displacement order C,B,A: V(A,B,C,C') + V(A,B,C',B') + V(A,B',C',A')

The volume of each tetrahedron is computed in accordance with the orientation of the facet (A,B,C). By adding all the new facet contributions, six times of the volume variation of the object P is obtained. If β_i is the sum of the cross products that are factors of the displacement dU_i, the expression of the volume variation of P is obtained:

$$6\Delta V(P) = \frac{1}{6}\sum_{i=1}^{n} \beta_i \cdot dU_i \tag{6.20}$$

The problem of volume preservation between object P and deformed object P' is to solve the equation $\Delta V(P) = 0$, i.e.,

$$\sum_{i=1}^{n} \beta_i \cdot dU_i = 0 \tag{6.21}$$

6.5.4 Volume Preservation of the Deformed Objects

In section 6.5.2, the deformation function d only depends on the matrix parameter ξ once the displacement constraints are set. ξ should be found to achieve the volume preservation of the object on which the deformation function d is applied. Equation (6.21) is rewritten into to bring the unknown ξ in evidence.

$$\sum_{i=1}^{n} \beta_i(\xi) \cdot dU_i(\xi) = 0 \tag{6.22}$$

Since the terms β_i's depend on the displaced vertices of the deformed object P', they are expressed as a function of ξ. This equation, cubic on the components of the matrix ξ of any dimension, accepts a set S_v of an infinity of solutions. Choosing a value of ξ among this infinity without a specific condition may lead to non-intuitive deformation with excessive expansion of the object after the volume correction or with an incoherent self-intersection in the resulting deformed object. The value of ξ in S_v is chosen to respect only an intuitive 'visual criterion'. This visual criterion indicates that the volume preserving deformed object should be the closest of the non-optimizing deformed object, i.e., ξ=0. Since the deformation function of DOGME is expressed linearly on ξ, the evaluation of a distance between two deformed objects is strongly related to ξ. The following 'visual criterion' is taken:

$$\|\xi_{\min}\| = \min_{\xi \in Sv}(\|\xi\|) \tag{6.23}$$

with the norm of the matrix ξ defined by $\|\xi\| = \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{3} \xi_{ij}^2}$, where ξ_{ij}'s are the components of the matrix ξ. Formally, the following problem is to be solved:

$$\|\xi_{\min}\| = \min_{\xi \in Sv}(\|\xi\|) \text{ with } S_v = \left\{\xi \in R^{m\cdot 3} / \sum_{i=1}^{n} \beta_i(\xi) \cdot dU_i(\xi) = 0\right\} \tag{6.24}$$

The problem expressed in this equation is easy to solve by an iterative approximation. Equation (6.22) is difficult to solve because it is of degree three on the components of the matrix ξ that is of any dimension. If the terms $\beta_i(\xi)$'s are constant with ξ, then equation (6.22) becomes linear and equation (6.24) is easy to solve. In order to solve the problem, β_i is successively approximated. The first step of iteration consists in calculating the terms $\beta_i(\xi)$ that appear in equation (6.22) at $\xi=0$. The displaced vertices of the polyhedron P' that occur in this calculation are given by the deformation function d taken at $\xi=0$. $\beta_i^{(1)} = \beta_i\ (\xi=0)$ represents the values of β_i for this first step. $\beta_i^{(1)}$ is constant and equation (6.22) is rewritten as follows:

$$\sum_{i=1}^{n} \beta_i^{(1)} \cdot dU_i(\xi) = 0 \tag{6.25}$$

Since the displacement dU_i is expressed linearly on ξ, equation (6.25) is a linear equation of a plane parameterized by the components of the matrix ξ. Suppose that $\beta_{ij}^{(1)}$ is the j^{th} coordinate of $\beta_i^{(1)}$. By expanding the scalar product in equation (6.25), the equation to be solved becomes:

$$\sum_{i=1;j=1..3}^{n} \beta_{ij} f'(U_i)(X^+ D_j + (I - X^+ X)\xi_j) = 0 \tag{6.26}$$

The plane equation is obtained by this equation:

$$\sum_{k=1;j=1..3}^{n} A_{kj}\xi_{kj} = B \tag{6.27}$$

with $A_{kj} = \sum_{i=1}^{n} \beta_{ij} f'(U_i)(I - X^+ X)_k$ where $(I\text{-}X^+X)_k$ is the k^{th} column of $(I\text{-}X^+X)$ and $B = -\sum_{i=1;j=1..3}^{n} \beta_{ij} f'(U_i) X^+ D_j$. To satisfy the condition ξ closest to zero, it is needed to

project the origin onto the plane. According to equation (6.27), the projection of the origin is given by:

$$\xi_{kj} = \frac{BA_{kj}}{\sum_{i=1;l=1..3}^{m} A_{il}^2}, \text{ with } k \in [1,m] \text{ and } j \in [1,3]$$

This iterative method continues until ξ provides a volume variation close to zero. The volume preserving surface warping algorithm is as follows [Aubert and Bechmann, 1997]:

(1) Extrusion function f of dimension m is given. Initial polyhedron P with its n vertices $(P_i)_{i\in[1,n]}$ is given;

(2) Set displacements constraints; deduce X and D from displacement constraints and f. Compute pseudo-inverse X^+;

(3) The aim is to find the deformed object P' that preserves the volume.

Set *iter* = 0
Set $\xi^{new} = 0$
Repeat
$\xi^{old} \leftarrow \xi^{new}$
Compute P' = *Deformation (P,* ξ^{old}*)*
Set $\beta_i = (\beta_{i1}, \beta_{i2}, \beta_{i3})$ to zero vector for each $i \in [1,n]$
For each triangular facet (P_i, P_j, P_k) belongs to P
CumulateBeta (β_i, β_j, β_k)
Set $B = -\sum_{i=1;j=1..3}^{n} \beta_{ij} f'(U_i) X^+ D_j$ for all $k \in [1,m]$ and $j \in [1,3]$
Set $A_{kj} = \sum_{i=1}^{n} \beta_{ij} f'(U_i)(I - X^+ X)_k$
Set $\xi_{kj}^{new} = \frac{BA_{kj}}{\sum_{i=1;l=1..3}^{m} A_{il}^2}$, for all $k \in [1,m]$ and $j \in [1,3]$
iter $\leftarrow$ *iter* + 1
Until ($\left\|\xi^{new} - \xi^{old}\right\| < d_{limit}$ or *iter* $\geq iter_{max}$
The preserved volume polydedron is obtained by P' = Deformation (P, ξ^{new}).

Deformation (P, ξ*):* Compute deformed polyhedron P' from P with the value ξ.

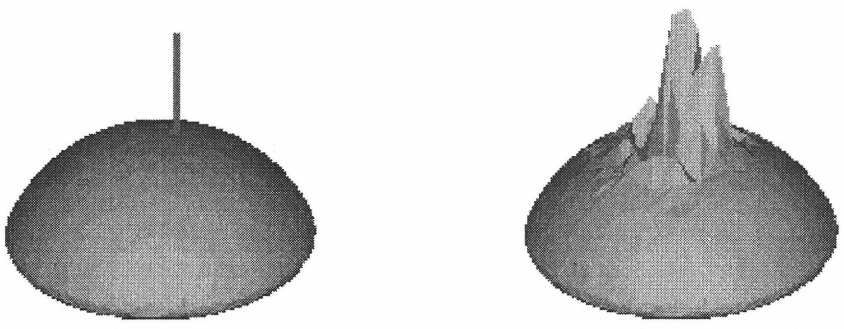
For each vertex P_i of P do

$$P_i' = P_i + f^t(P_i)(X^+D + (I - X^+X)\xi)$$

CumulateBeta (β_i, β_j, β_k): Note that (P_i, P_j, P_k) must respect the orientation of P (topological data must be included into the structure of the polyhedron). (P_i', P_j', P_k') is the deformed facet that belongs to P'.

$$\beta_i \leftarrow \beta_i + 2(P_iP_j \wedge P_jP_k) + P_iP_j' \wedge P_j'P_k + P_iP_j \wedge P_jP_k' + 2(P_iP_j' \wedge P_j'P_k')$$
$$\beta_j \leftarrow \beta_j + 2(P_iP_j \wedge P_jP_k) + P_i'P_j \wedge P_jP_k + P_iP_j \wedge P_jP_k' + 2(P_i'P_j \wedge P_jP_k')$$
$$\beta_k \leftarrow \beta_k + 2(P_iP_j \wedge P_jP_k) + P_i'P_j \wedge P_jP_k + P_iP_j' \wedge P_j'P_k + 2(P_i'P_j' \wedge P_j'P_k)$$

The following example is the implanted material deformation used in breast enhancement simulation (Figure 6.15). Figure 6.15(a) is the original object. The



(a) (b)

Figure 6.15 One example of volume preservation surface warping.

displacement vector is attached to the surface of the object. Figure 6.15(b) illustrates the deformation result using the volume-preserving surface warping algorithm. However the result is not good enough to be used. The surface around the displacement vector is not smooth after the deformation. The following possibilities may cause the problem:

(1) The vertex orientation of each triangle in the original model data set is not in the same order. This gives the incorrect volume variation.

(2) The extrusion function f and matrix X and D are fixed before the deformation. During the deformation process, these values do not reflect the object deformation, and thus cause large errors.

(3) The whole algorithm applies many approximations in several steps that may create non-smooth results.

According to the algorithm of [Aubert and Bechmann, 1997] and the experimental result in this work, the following preliminary solutions and ideas may be applied in the future research work.

The extrusion function f is set before the deformation and it applies global B-splines once in the implemented algorithm. In the future work, local B-splines may be applied to achieve specified area deformation with volume preservation. When the model is in the process of deforming, the B-splines parameters should change accordingly. Thus function f should also be changed.

There is an infinity of solutions to the problem of volume preserving surface warping. The value ξ among these solutions without a specific condition may lead to a non-intuitive deformation result. Other constraints such as geometrical and physical properties are free to add in order to choose a particular solution. In the application of breast enhancement simulation, the bounded deformation of each vertex can be added in the constraints. Some deformation tolerances based on the experience in order to achieve a visual criterion can also be considered.

As in the application of nose surface warping, users can manipulate a scale widget that defines the desired volume change format for a chosen primitive. The volume preservation algorithm can be repeatedly performed while the scale changes.

The use of a single displacement vector may lack enough information to define the exact object deformation. Additionally the convexity of the object is not kept well with one displacement constraint. More displacement constraints can be chosen according to the specific requirements of the deformation result.

6.6 Summary

This work has developed a direct manipulation of free-form deformation based method and application for pre-surgical planning. Users of this application need just to select one control point instead of many in the object space. This greatly facilitates the local 3D warping and makes it ideal for novice users of such applications, e.g., physicians and nurses. One displacement vector decides the final target point of the selected point. The user interface supplies a friendly interactive tool in the plastic surgery. However, it is a potential disadvantage to use only one control point. Several parameters can be altered to obtain various deformation results according to different patient noses. Lattice resolution can decide the deformation area. Displacement vector length and direction are used to change, for example, nose size and shape and the angle between the nose and the upper lip. One overall displacement can be performed in several steps with different lattice resolution and vector direction in each step. The volume change of a face model is analyzed and specified. A simplified skin-muscle model is also proposed. With this model, the implanted material form and size on a facial model is simulated and shown.

The example results have shown that the nose face is smoothly deformed. The application is very fast at the level of milliseconds and thus suitable for real-time operation. Besides the nose augmentation applications in plastic surgery, the application can be used in entertainment industry.

Currently, the proposed method and application do not consider the actual skull form and the size of available implanted material. Future work is needed to make the tool more useful. This work proposes one initial yet important step in pre-surgery planning. To the authors' knowledge, the proposed method represents the simplest way that can be conceived to achieve desired 3D warping result that was never possible before without using sophisticated methods, such as [Lee et al., 1999].

The volume-preserving space deformation algorithm proposed in [Aubert and Bechmann, 1997] is introduced and applied in this work. However the surface deformation result is not smooth in the experimental example. The possible reasons that cause non-smooth deformation are analyzed and several future research directions on this research are specified.

CHAPTER 7

CONCLUSIONS

This dissertation develops 2D image segmentation algorithms in solar images to detect filament disappearance automatically. One mesh simplification algorithm in 3D image is described. The compression results with proposed algorithm are compared with other major simplification algorithms. This work also applies 3D warping technique in plastic surgery planning application.

7.1 Summary of Contributions

The contributions of this dissertation are summarized into three aspects:

(1) Image segmentation in solar filament disappearance application

One efficient filament detection algorithm is developed. It combines thresholding and region-growing methods. Based on the accurate detection of filaments, the disappearing filaments are reported in consecutive images. The images in 1999 are analyzed with this system and three statistical results are obtained. The results match well with those performed by pre-researchers using manual methods.

(2) 3D mesh simplification

The second contribution of this dissertation is to develop a 3D mesh simplification. The algorithm uses Root Mean Square (RMS) distance error metric to decide the facet curvature. Two vertices of one edge and the surrounding vertices decide the average plane. The simplification results are excellent and the computation speed is fast. The algorithm is compared with six other major simplification algorithms.

(3) 3D image warping in pre-surgery simulation

This work develops a direct-manipulation-of-free-form-deformation-based method and application for pre-surgical planning. The developed user interface provides a friendly interactive tool in the plastic surgery. Nose augmentation surgery is presented as an example. Displacement vector, lattice resolution are used to obtain various deformation results. During the deformation, the volume change is also considered based on the simplified skin-muscle model.

7.2 Limitations

This research also has some limitations:

(1) In solar filament detection program, the images used in the program from BBSO are obtained from camera. There exist some errors during the procedures from the original solar image to the full-disk Hα images in this work. The detection results may be inaccurate because of these errors.

(2) High resolution solar images are available in BBSO database. However this work processes 1024 × 1024 images only due to the computer capability and speed requirements.

(3) The pre-surgery simulation uses one displacement vector in this work. In actual application, the plastic surgeon is not able to plan one ideal surgical treatment only with a single displacement vector. The tissue model proposed includes fat and musle tissue. But in the application of a rhinoplasty, at the tip of the nose, the muscle model should include cartilage, fat and skin instead of muscle and fat.

7.3 Future Research

There are several ways in which this work could be extended in the future. However, the following avenues appear particularly important and promising.

(1) The proposed solar filament detection can be adjusted to detect prominences. It is preferable to process high resolution images to detect the locations of filaments more accurately.

(2) The proposed mesh simplification algorithm can be improved performance on very large data sets with more than one million triangles. More conditions should be set up to simplify the huge data set without difficulty.

(3) In proposed pre-surgery simulation, the exact nose skull structure is not considered. To achieve more accurate simulation result, it is necessary to obtain skull structure. The 3D nose skull structure can be extracted from a series of 2D head slide images using interpolation techniques.

(4) The 3D warping in pre-surgery application is a convenient tool for doctors. However more effective methods in user interface can be improved. One displacement vector is not enough in actual surgery planning. How to add more control vectors without adding inconvenience to doctors is another direction for future work.

REFERENCES

1. B. Achermann, R. Peleg, X. Jiang, H. Bunke, D. Feinendegen, Y. Bruhlmann and H. Tschopp, "Computer-based assistance of surgeons in the judgment of plastic nose surgeries (Rhinoplasty)," *Proceedings of IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 21-26, 1997.

2. J. Aggarwal and N. Nandhakumar, "On the computation of motion from sequences of images – a review," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 917-935, 1998.

3. N. Ahuja, "Multiscale image segmentation by integrated edge and region detection," *IEEE Trans. On Image Processing*, vol. 6, no. 5, pp. 642-654, 1997.

4. M. Algorri, F. Schmitt, "Mesh simplification," *Computer Graphics Forum*, vol. 15, no. 3, 1996.

5. E. Angel, *Interactive Computer Graphics, A Top-Down Approach with OpenGL, Second Edition,* Addison Wesley Longman, Inc., 2000.

6. F. Arman and J. Aggarwal, "Model-based object recognition in dense-range images – a review," *ACM Computing Survey*, vol. 25, no. 1, pp. 5-43, 1993.

7. F. Aubert and D. Bechmann, "Volume-preserving space deformation," *Computer and Graphics*, vol. 21, no. 5, pp. 625-639, 1997.

8. A. Barr, "Global and local deformation of solid primitives," *Computer Graphics*, vol. 18, no. 3, pp. 21-30, 1984.

9. G. Baxes, *Digital Image Processing: Principles and Application.* John Wiley & Sons, Inc., New York, 1994.

10. D. Bechmann, "Space deformation models survey," *Computer & Graphics*, vol. 18, no. 4, pp. 571-586, 1994.

11. T. Beier, S. Neely, "Feature-based image metamorphosis," *Proceedings of SIGGRAPH 92*, pp. 35-42, 1992.

12. R. Bergevin, D. Laurendeau, D. Poussart, "Registering range views of multipart objects," *Computer Vision and Image Understanding*, vol. 61, no. 1, pp. 1-16, 1995.

13. R. Bergevin, M. Soucy, H. Gagnon and D. Laurendeau, "Towards a general multi-view registration technique," *IEEE Trans. On Pattern Recognition and Machine Intelligence,* vol. 18, no. 5, pp. 540-547, 1996.

14. P. Besl and R. Jain, "Three-dimensional object recognition," *Computer Survey*, vol. 17, no. 1, pp. 75-141, 1985.

15. P. Besl and N. Mckay, "A method for registration of 3-D shapes," *IEEE Trans. On Pattern Recognition and Machine Intelligence*, vol. 14, no. 2, pp. 239-256, 1992.

16. P. Besl and R. Jain, "Segmentation through variable-order surface fitting," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 10, pp. 167-192, 1988.

17. R. Beveridge, "Segmenting images using localized histograms and region merging," C*omputer Vision, Graphics, Image Processing*, vol. 2, pp. 311-347, 1989.

18. W. Birkfellner, K. Huber, A. Larson, D. Hanson, M. Diemling, P. Homolka and H. Bergmann, "A modular software system for computer-aided surgery and its first application in oral implantology," *IEEE Trans. on Medical Imaging*, vol. 19, no. 6, pp. 616-620, 2000.

19. G. Blais and M. Levine, "Registering multiview range data to create 3D computer objects," *IEEE Trans. On Pattern Recognition and Machine Intelligence*, vol. 17, no. 8, pp. 820-824, 1995.

20. V. Blanz and T. Vetter, "A morphable model for the synthesis of 3D faces," *Proceedings of SIGGRAPH 99*, pp. 187-194, 1999.

21. R. Bogart and T. Bai, "Confirmation of a 152 day periodicity in the occurrence of solar flares inferred from microwave data," *Astrophysical Journal,* L 51, pp. 299-310, 1985.

22. R. Bolle and B. Vemuri, "On three-dimension surface reconstruction methods," *IEEE Trans. On Pattern Recognition and Machine Intelligence*, vol. 13, no. 1, pp. 1-13, Jan. 1991.

23. P. Borrel and A. Rappoport, "Simple constrained deformations for geometric modeling and interactive design," *ACM Transactions on Graphics*, vol. 13, no. 2, pp. 137-155, 1994.

24. L. Brown, "A survey of image registration techniques," *ACM Computing Surveys*, vol. 24, no. 4, pp. 325-376, 1992.

25. V. Cappellini, L. Alparone, G. Galli, P. Lange, A. Mecocci, L. Menichetti, G. Capanni and R. Carla, "Digital processing of stereo images and 3-D reconstruction techniques," *International Journal on Remote Sensing*, vol. 12, no. 3, pp. 477-490, 1991.

26. K. R. Castleman, *Digital Image Processing*, Prentice Hall Press, 1996.

27. D. Chen, A. State and S. Banks, "Interactive shape metamorphosis," *Symposium on interactive 3D graphics*, pp. 43-44, 1995.

28. S. Chen, W. Lin, and C. Chen, " Split-and-merge image segmentation based on localized feature analysis and statistical tests," *Computer Vision, Graphics, Image Processing: Graph, Models Image processing*, vol. 53, pp. 457-475, 1991.

29. Y. Chen and G. Medioni, "Object modelling by registration of multiple range images," *Image and Vision Computing*, vol. 10, no. 3, pp. 145-155, April 1992.

30. C. Chu and J. K. Aggarwal, "The integration of image segmentation maps using region and edge information," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 15, pp. 241-1252, Dec. 1993.

31. A. Ciampalini, P. Cignoni, C. Montani and R. Scopigno, "Multiresolution decimation based on global error," *The visual Computer*, vol. 13, pp. 228-246, 1997.

32. P. Cignoni, C. Montani, R. Scopigno, "A comparison of mesh simplification algorithms," *Computer and Graphics*, vol. 22, no. 1, pp. 37-54, 1998.

33. P. Cignoni, C. Rocchini and R. Scopigno, "Metro: measuring error on simplified surfaces," *Computer Graphics Forum*, vol. 17, no. 2, pp. 167-174, June 1998.

34. J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, W. Wright, "Simplification envelopes," *SIGGRAPH '96 Proceedings*, pp. 119-128, 1996.

35. D. Cohen-Or, D. Levin, A. Solomovici, "Three dimensional distance field metamorphosis," *ACM Transactions on Graphics*, vol. 17, no. 2, pp. 116-141, 1998.

36. S. Coquillart, "Extended free-form deformation: a sculpturing tool for 3D geometric modeling," *Computer Graphics*, vol. 24, no. 4, pp. 187-196, 1990.

37. S. Coquillart and P Jancène, "Animated free-form deformation: an interactive animation technique," *Computer Graphics*, vol. 25, no. 4, pp. 23-26, 1991.

38. N. Crosby, M. Aschwaden and B. Dennis, "Frequency distributions and correlations of solar X-lay flare parameters," *Solar Physics*, vol. 143, pp. 275-299, 1993.

39. B. Curless and M. Levoy, "A volumetric method for building complex models from range images," *Proceedings of SIGGRAPH '96*, pp. 303-312, 1996.

40. D. DeCarlo and J. Gallier, "Topological evolution of surfaces," *Graphics Interface '96*, Toronto, Canada, pp. 194-203, 1996.

41. H. Delingette, Y. Watanabe and Y. Suenaga, "Simplex based animation," *Computer Animation '93*, Computer Graphics International Geneva, Switzerland, pp. 13-28, 1993.

42. C. Denker, A. Johannesson, W. Marquette, P. Goode, H. Wang and H. Zirin, "Synoptic Hα full-disk observations of the sun from BBSO," *Solar Physics*, vol. 184, no. 1, pp. 87-102, 1999.

43. C. Dorai, G. Wang, A. Jain, C. Mercer, "Registration and integration of multiple object views for 3D model construction," *IEEE Trans. On Pattern Recognition and Machine Intelligence*, vol. 20, no.1, pp. 83-89, January 1998.

44. C. Dumitrache, "On the evolution of filaments," *Solar Physics*, vol. 173, pp. 281-304, 1997.

45. J. Duncan and N. Ayache, "Medical image analysis: progress over two decades and the challenges ahead," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 85-106, 2000.

46. M. Eck, T, DeRose, T. Duchamp, H. Hoppe, M. Lounsbery and W. Stuetzle, "Multiresolution analysis of arbitrary meshes," *SIGGRAPH '95 Proceedings*, pp. 173-182, 1995.

47. C. Erikson, "Polygonal simplification: an overview," *Technical Report TR96-016*, University of North Carolina, 1996.

48. O. Faugeras, *Three-dimensional Computer Vision: A Geometric Viewpoint*, The MIT Press, Cambridge, MA, USA, 1993.

49. J. Feng, P. Heng and T. Wong, "Accurate B-spline free-form deformation of polygonal objects," *Journal of Graphics Tools*, vol. 3, no. 3, pp. 11-27, 1998.

50. D. Forsey and R. Bartels, "Hierarchical B-spline refinement," *Computer Graphics*, vol. 22, no. 4, pp. 205-212, 1988.

51. P. Gao and T. Sederberg, "A work minimization approach to image morphing," *The Visual Computer*, vol. 14, pp. 390-400, 1998.

52. M. Garland, P. Heckbert, "Surface simplification using quadric error metrics," *SIGGRAPH '97 Proceedings*, pp. 209-216, 1997.

53. M. Garland, *Quadric-based polygonal surface simplification*, Ph.D. dissertation, School of Computer Science, Carnegie Mellon University, May 1999.

54. J. Gomes, L. Darsa, B. Costa and L. Velho, *Warping and Morphing of Graphical Objects*, Morgan Kaufmann Publishers, Inc., San Francisco, California, 1998.

55. A. Goshtasby, "Three-dimensional model construction from multiview range images: survey with new results," *Pattern Recognition*, vol. 31, no. 11, pp. 1705-1714, 1998.

56. A. Gregory, A. State, M. Lin, D. Manocha and M. Livingston, "Feature-based surface decomposition for correspondence and morphing between polyhedra," *Proceedings of Computer Animation '98*, 1998.

57. A. Gueziec, "Surface simplification inside a tolerance volume," *Technical Report, IBM Research Report*, Yorktown Heights, NY 10598, March 1996.

58. L. Guibas, J. hershberger, "Morphing simple polygons," *Proceedings of 10th Computational Geometry*, pp. 267-276,1994.

59. L. Gyori, "Automation of area measurement of sunspots," *Solar Physics*, vol. 180, pp. 109-130, 1998.

60. R. Haralick and L. Shapiro, "Survey: Image segmentation techniques," *Computer Vision, Graphics, Image Processing*, vol. 29, no. 1, pp. 100-132, 1985.

61. K. Haris, S. Efstratiadis, N. Maglaveras, and A. K. Katsaggelos, " Hybrid image segmentation using watersheds and fast region merging," *IEEE Trans. on Image Processing*, vol. 7, no. 12, pp. 1684-1699, Dec. 1998.

62. T. He, S. Wang and A. Kaufman, "Wavelet-based volume morphing," *Proceedings of Visualization '94*, Washington, DC, pp. 85-92, 1994.

63. D. Hearn and M. Baker, *Computer Graphics, C Version*, Prentice Hall, Inc., 1997.

64. P. Heckbert, M. Garland, "Survey of polygonal surface simplification algorithms," *SIGGRAPH '97 course notes #25, ACM SIGGRAPH*, 1997.

65. P. V. Henstock and D. M. Chelberg, "Automatic Gradient Threshold Determination for edge detection," *IEEE Trans. on Image Processing*, vol. 5, no. 5, pp. 784-787, May 1996.

66. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Surface reconstruction from unorganized points," *Computer Graphics (SIGGRAPH '92 Proceedings)*, vol. 26, no. 2, pp. 71-78, July 1992.

67. H. Hoppe, T. DeRose, T. Duchamp, J. McDonald and W. Stuetzle, "Mesh Optimization," *SIGGRAPH '93 Proceedings*, pp. 19-26, 1993.

68. H. Hoppe, *Surface reconstruction from unorganized points*, Ph.D. Dissertation, Department of Computer Science and Engineering, University of Washington, June 1994.

69. H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, W. Stuetzle, "Piecewise smooth surface reconstruction," *SIGGRAPG '94 Proceedings*, pp. 295-302, July 1994.

70. H. Hoppe, "Progressive meshes," *SIGGRAPH '96 Proceedings*, pp. 99-108, Aug. 1996.

71. B. Horn, "Closed form solution of absolute orientation using unit quaternions," *Journal of Optical Society A*, vol. 4, no. 4, pp. 629-642, April 1987.

72. W. Hsu, J. Hughes and H. Kaufman, "Direct Manipulation of free-form deformations," *Computer Graphics*, vol. 26, no. 2, pp. 177-184, 1992.

73. T. Huang and S. Blostein, "Robust algorithms for motion estimation based on two sequential stereo image pairs," *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pp. 518-523, June 1985.

74. J. Hughes, "Scheduled Fourier volume morphing," *Proceedings of SIGGRAPH 92*, pp. 43-46, 1992.

75. A. Johnson, M. Hebert, "Control of polygonal mesh resolution for 3-D computer vision," *Technical Report CMU-RI-TR-96-20*, Carnegie Mellon University, 1997.

76. R. Jones, *Introduction to MFC Programming with Visual C++*, Prentice Hall PTR, Upper Saddle River, NJ, USA, 2000.

77. T. Kanai, H. Suzuki and F. Kimura, "Three-dimensional geometric metamorphosis based on harmonic maps," *The Visual Computer*, vol. 14, no. 4, pp. 166-176, 1998.

78. J. Kent, W. Carlson, R. Parent, "Shape transformation for polyhedral objects," *Proceeding of SIGGRAPH 92,* pp. 47-54, 1992.

79. L. Kitchen and A. Rosenfeld, "Edge evaluation using local edge coherence," *IEEE Trans. on System Man, Cybernetics*, vol. SMC-11, no.9, pp. 597-605, Sept. 1981.

80. R. Klein, G. Liebich, W. StraBer, "Mesh reduction with error control," *Proceedings of the conference on visualization 96*, pp. 311-318, 1996.

81. R. Koch, M. Gross, F. Carls, D. Buren, G. Fankhauser and Y. Parish, "Simulating facial surgery using finite element models," *Proceedings of the 23rd Annual Conference on Computer Graphics*, pp. 421-428, 1996.

82. P. A. Laplante and A. D. Stoyenko, *Real-time Imaging*. IEEE Press, 1996.

83. F. Lazarus and A. Verroust, "Feature-based shape transformation for polyhedral objects," *The 5th Eurographics Workshop on Animation and Simulation*, pp. 1-14, 1994.

84. F. Lazarus and A. Verroust, "Metamorphosis of cylinder-like objects," *International Journal on Visualization Computer Animation*, vol. 8, pp. 131-146, 1997.

85. F. Lazarus, A. Verroust, "Three-dimensional metamorphosis: a survey," *The Visual Computer*, vol. 14, pp. 373-389, 1998.

86. A. Lee, W. Sweldens, P. Schroder, L. Cowsar, D. Dobkin, "MAPS: Multiresolution adaptive parameterization of surfaces," *Proceedings of SIGGRAPH* 98, pp. 95-104, 1998.

87. A. Lee, D. Dobkin, W. Sweldens, P. Schroder, "Multiresolution Mesh Morphing," *Proceedings of ACM SIGGRAPH 99*, pp. 343-350, 1999.

88. S. Lee, K. Chwa, S. Shin, G. Wolberg, "Image metamorphosis using snakes and free-form deformations," *Proceedings of SIGGRAPH 95*, pp. 439-448, 1995.

89. S. Lee, K. Chwa, J. Hahn and S. Shin, "Image morphing using deformation techniques," *Journal of Visualization Computer Animation*, vol. 7, pp. 3-23, 1996.

90. T. Lee, Y. Sun, Y. Lin, L. Lin and C. Lee, "Three-dimensional facial model reconstruction and plastic surgery simulation," *IEEE Trans. on Information Technology in Biomedicine*, vol. 3, no. 3, pp. 214-220, 1999.

91. Y. Lee, D. Terzopoulos and K. Waters, "Realistic modeling for facial animation," *Proceedings of SIGGRAPH '95*, pp. 55-62, 1995.

92. A. Lerios, C. Garfinkle, M. Levoy, "Feature-based volume metamorphosis," *Proceedings of ACM SIGGRAPH 95*, pp. 449-456, 1995.

93. Z. Lin, H. Lee and T. Huang, "Finding 3-D point correspondences in motion estimation," *Proceedings of 8th International Conference on Pattern Recognition*, pp. 303-305, Oct. 1986.

94. P. Lindstrom and G. Turk, "Fast and memory efficient polygonal simplification," *Proceedings of the conference on visualization '98*, pp. 279-286, 1998.

95. S. Lippman and J. Lajoie, *C++ Primer, Third Edition*, Addison Wesley Longman, Inc., Reading, MA, USA, 1998.

96. K. Low, T. Tan, "Model simplification using vertex-clustering," *1997 Symposium on Interactive 3D Graphics*, pp. 75-81, 1997.

97. R. MacCracken and K. Joy, "Free-form deformations with lattices of arbitrary topology," *Proceeding of SIGGRAPH '96*, pp. 181-188, 1996.

98. D. Marr, *Vision,* Freeman, 1982.

99. S. F. Martin, "Conditions for the formation and maintenance of filaments," *Solar Physics*, vol. 182. no. 1, pp. 107-137, 1998.

100. T. Masuda, N. Yokoya, "A robust method for registration and segmentation of multiple range images," *Computer Vision and Image Understanding*, vol. 61, no. 3, pp. 295-307, 1995.

101. G. F. Mclean and M. E. Jernigan, "Hierarchical edge detection," *Computer Vision, Graphics, Image Processing*, vol. 44, pp. 350-366, 1988.

102. V. S. Nalwa and T. O. Binford, "On detecting edges," *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI-8, no. 6, pp. 699-714, Nov. 1986.

103. N. Pal and S. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, pp. 1277-1294, 1993.

104. R. Parent, "Shape transformation by boundary representation interpolation: a recursive approach to establishing face correspondences," *Journal of Visualization and Computer Animation*, vol. 3, pp. 219-239, 1992.

105. N. Papamarkos and B. Gatos, "A new approach for multilevel threshold selection," *Computer Vision, Graphics, Image Processing: Graphical Models Image Processing*, vol. 56, no. 5, pp. 357-370, 1994.

106. T. Peters, B. Davey, P. Munger, R. Comeau, A. Evans and A. Olivier, "Three-dimensional multimodal image-guidance for neurosurgery," *IEEE Trans. on Medical Imaging*, vol. 15, no. 2, pp. 121-128, 1996.

107. A. Rappoport, A. Sheffer and M. Bercovier, "Volume-preserving free-form solids," *IEEE Trans. on Visualization and Computer Graphics*, vol. 2, no. 1, pp. 19-27, 1996.

108. D. Rueckert, L. Sonoda, C. Hayes, D. Hill, M. Leach and D. Hawkes, "Nonrigid registration using free-form deformation: application to breast MR images," *IEEE Trans. on Medical Imaging*, vol. 18, no. 8, pp. 712-721, 1999.

109. D. Ruprecht, H. Muller, "Image warping with scattered data interpolation," *IEEE Computer Graphics and Applications*, pp. 37-43, March 1995.

110. J. C. Russ, *The Image Processing Handbook*, CRC Press, 1992.

111. P. K. Sahoo, S. Soltani and A. K. C. Wong, "A survey of thresholding techniques," *Computer Vision, Graphics and Image Processing*, vol. 41, pp. 233-260, 1988.

112. Y. Sato, M. Nakamoto, Y. Tamaki, T. Sasama, I. Sakita, Y. Nakajima, M. Monden and S. Tamura, "Image guidance of breast cancer surgery using 3-D ultrasound images and augmented reality visualization," *IEEE Trans. on Medical Imaging*, vol. 17, no. 5, pp. 681-693, 1998.

113. W. Schroeder, J. Zarge and W. Lorensen, "Decimation of triangle meshes," *Computer Graphics (SIGGRAPH '92 Proceedings)*, vol. 26, no. 2, pp. 65-70, July 1992.

114. T. Sederberg and S. Parry, "Free-form deformation of solid geometric models," *Computer Graphics*, vol. 20, no. 4, pp. 151-160, 1986.

115. T. Sederberg and E. Greenwood, "A physically based approach to 2-D shape blending," *Computer Graphics*, vol. 26, no. 2, pp. 25-34, 1992.

116. T. Sederberg, P. Gao, G. Wang and H. Mu, "Shape blending: an intrinsic approach to the vertex path problem," *Computer Graphics Conference Series*, pp. 15-18, 1993.

117. S. Seitz, C. Dyer, "View Morphing," *Proceedings of SIGGRAPH 96*, pp. 21-30, 1996.

118. D. Smythe, "A two-pass mesh warping algorithm for object transformation and image interpolation," *Technical Report 1030*, ILM Computer Graphics Department, Lucasfilm, San Rafael, California, 1990.

119. M. Soucy, D. Laurendeau, "A general surface approach to the integration of a set of range views," *IEEE Trans. On Pattern Recognition and Machine Intelligence*, vol. 17, no. 4, pp. 344-358, April 1995.

120. M. Soucy, D. Laurendeau, "Multiresolution surface modeling based on hierarchical triangulation," *Computer Vision and Image Understanding*, vol. 63, no. 1, pp. 1-14, Jan. 1996.

121. P. St-Jean, A. Sadikot, L. Collins, D. Clonda, R. kasrai, A. Evans, and T. Peters, "Automated atlas integration and interactive three-dimensional visualization tools for planning and guidance in functional neurosurgery," *IEEE Trans. on Medical Imaging*, vol. 17, no. 5, pp. 672-680, 1998.

122. B. Stroustrup, *The C++ Programming Language, Third Edition*, Addison Wesley Longman, Inc., Reading, MA, USA, 1997.

123. V. Torre and T. A. Poggio, "On edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no.2, pp. 147-163, 1986.

124. G. Turk, "Re-Tiling polygonal surfaces," *Computer Graphics (SIGGRAPH '92 Proceedings)*, vol. 26, no. 2, pp. 55-64, July 1992.

125. G. Turk, M. Levoy, "Zippered polygon meshes from range images," *SIGGRAPH '94 Proceedings*, pp. 311-318, 1994.

126. S. Venkatesh and L. J. Kitchen, "Edge evaluation using necessary components," *Computer Vision, Graphics, Image Processing: Graphic, Models Image Processing*, vol. 54, pp. 23-30, 1992.

127. H. Wang, A. E. Komenda, F. Tand and H. Zirin, "Filament disappearances during the period of September 1991 through September 1994," *Solar Physics*, vol. 178, no. 1, pp. 109-117, 1998.

128. M. A. Wani, B. G. Batchelor, "Edge-region-based segmentation of range images," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, pp. 314-319, Mar. 1994.

129. J. S. Weszka and A. Rosenfeld, "Threshold evaluation technique," *IEEE Trans. Systems, Man, Cybernetics*, vol. 8, no. 8, pp. 622-629, 1978.

130. G. Wolberg, *Digital Image Warping*, IEEE Computer Society Press, Los Alamitos, California, 1990.

131. G. Wolberg, "Image morphing, a survey," *the Visual Computer*, vol. 14, pp. 360-373, 1998.

132. M. Woo, J. Neider, T. Davis and D. Shreiner, *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*, Addison-Wesley, Reading, MA, USA, August 1999.

133. J. Xia, A. Varshney, "Dynamic view-dependent simplification for polygonal models," *Proceedings of Visualization 96*, pp. 327-334, 1996.

134. Z. Zhang, "Iterative point matching for registration of free-form curves and surfaces," *International Journal of Computer Vision*, vol. 13, no. 2, pp. 119-152, 1994.

135. D. Zhao and X. Zhang, " Range-data-based object surface segmentation via edges and critical points," *IEEE Trans. On Image Processing*, vol. 6, no. 6, pp. 826-830, 1997.