RECONSTRUCTION OF A 3D HUMAN HEAD MODEL FROM IMAGES

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

REZA ZARE HASSANPOUR

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

DOCTOR OF PHILOSOPHY

IN

THE DEPARTMENT OF COMPUTER ENGINEERING

August 2003

Approval of the Graduate School of Natural and Applied Sciences.

Prof. Dr. Canan Özgen
Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of DOCTOR OF PHILOSOPHY.

Prof. Dr. Ayşe Kiper
Head of Department

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of DOCTOR OF PHILOSOPHY.

Assoc.prof. Dr. Volkan Atalay
Supervisor

Examining Committee Members

Prof. Dr. Ayşe Kiper

Prof. Dr. Fatoş T. Yarman Vural

Prof. Dr. Turhan Alper

Assoc. Prof. Dr. Veysi İşler _____

Assoc. Prof. Dr. Volkan Atalay _____

ABSTRACT

RECONSTRUCTION OF A 3D HUMAN HEAD MODEL FROM IMAGES

Hassanpour, Reza Zare

Ph.D., Department of Computer Engineering

Supervisor: Assoc.prof. Dr. Volkan Atalay

August 2003, 90 pages

The main aim of this thesis is to generate 3D models of human heads from uncalibrated images. In order to extract geometric values of a human head, we find camera parameters using camera auto calibration. However, some image sequences generate non-unique (degenerate) solutions. An algorithm for removing degeneracy from the most common form of camera movement in face image acquisition is described. The geometric values of main facial features are computed initially. The model is then generated by gradual deformation of a generic polygonal model of a head. The accuracy of the models is evaluated using ground truth data from a range scanner. 3D models are covered with cylindrical texture values obtained from images. The models are appropriate for animation or identification applications.

Keywords: camera auto calibration, degenerate motion, human face modeling, Delaunay triangulation

ÖZ

GÖRÜNTÜLERDEN 3B INSAN KAFASI MODELI GERIÇATIMI

Hassanpour, Reza Zare

Doktora, Bilgisayar Mühendisliği Bölümü

Tez Yöneticisi: Assoc.prof. Dr. Volkan Atalay

Ağustos 2003, 90 sayfa

Bu tezin başlıca amacı ayarlanmamış görüntülerden üç boyutlu insan kafası modelleri üretmektir. Bir insan kafasının geometrik değerlerini çıkartabilmek amacıyla öncelikle kamera parametreleri kendi kendine ayarlama ile bulunmaktadır. Ancak, bazı görüntü dizileri birden fazla çözüm üretmektedir. Yüz görüntüleri alınırken bu tür sorunu ortadan kaldıran bir algoritma sunulmaktadır. Yüz özelliklerinin geometrik değerleri öncelikle hesaplanmakta ve ardından kafa modeli jenerik bir poligona dayalı kafa modelinin deformasyonu ile elde edilmektedir. Üretilen modellerin doğruluğu bir derinlik tarayıcıdan alınan yer bilgileri kullanılarak degerlendirilmektedir. 3B modeller görüntülerden çıkartılan silindirik doku değerleriyle de kaplanmaktadır. Modeller canlandırma ve kimlik belirleme uygulamalarında kullanıma uygundur.

Anahtar Kelimeler: kamera kalibrasyonu, Dejenere hareket, insan kafasini modelleme, Delaunay triangulasyonu

ACKNOWLEDGMENTS

I would like to express my deep gratitude to my advisor Dr. Volkan Atalay for providing an exciting and friendly working environment and also for his valuable and fruitful guidance, to Prof. Fatoş Vural for providing the opportunity of this development, to Prof. Turhan Alper for his understanding and encouragements. I am also grateful to the other members of the jury, Prof. Ayşe Kiper and Dr. Veysi İşler, who accepted this task with enthusiasm. And finally I would like to thank my family for their patience and support.

To my family

TABLE OF CONTENTS

CHAPTER

LIST OF FIGURES

LIST OF TABLES

NOTATIONS

Throughout the thesis bold capital letters are used to represent a matrix (e.g. **A**). Small bold letters represent vectors (e.g. **t**). Scalars and constant values are given using non-bold italic letters (e.g. s).

$\mathbb{P}^n$	Projective n-space
$\mathbf{P}$	Camera projection matrix
π_∞	Plane at infinity
$\mathbf{H}$	Homography matrix
$\mathbf{C}$	A Conic
$\mathbf{C}^*$	Dual of the Conic $\mathbf{C}$
$\mathbf{Q}$	A quadric
$\mathbf{Q}^*$	Dual of a quadric
$\mathbf{Q}$	A quadric
ω_∞	Absolute conic
Ω^*_∞	Absolute dual quadric
$\mathbf{R}$	A rotation matrix
$\mathbf{K}$	Intrinsic camera parameters matrix
f	Focal length of camera
λ	aspect ratio
γ	skew
$\mathbf{F}$	Fundamental matrix
$\mathbf{E}$	Essential matrix
$\mathbf{e}$	epipole
π_e	epipolar plane
RMS	Root Mean Square
$\parallel \mathbf{A} \parallel$	Frobonies norm of matrix $\mathbf{A}$

CHAPTER 1

Introduction

1.1 Motivation

There is a significant body of work on three dimensional (3D) analysis of images in recent years. Many application areas such as computer animation, medical imaging and visualization, and teleconferencing require 3D information of the environment that can be simulated on our machines. The first step in 3D analysis is modeling objects. This step, in general has been proven to be a difficult task to accomplish. The reason for the difficulty is the need for very sensitive and reliable measurements that can either be obtained by means of complicated measuring devices such as range scanners to find the depth information or by developing algorithms to extract this information from two dimensional (2D) images. In addition, these measurements are very sensitive to the environmental parameters such as lighting and noise. Furthermore, intrinsic complexity available in natural and deformable objects can intensify the problem. To deal with these problems most researchers have restricted the 3D objects to be modeled to a limited class of natural or artificial entities. This restriction makes possible to incorporate the knowledge about the class of objects into the system.

One of the 3D modeling applications is the identification, synthesis and ani-

mation of human faces. By human face, we mean the frontal part of the head including facial features, but in modeling sometimes we need other parts of the head such as hair. Nevertheless, in this manuscript we use head and face interchangeably. Most of the face recognition, expression understanding and animation algorithms developed so far work on 2D images. However, the use of a 2D model limits the range of face orientations that can be handled. For large variations of face orientation, 3D properties of the face is needed to be modeled. Many popular face recognition techniques operate on 2D face images and their performance can deteriorate quickly as a result of light variations and geometric transformations (scaling and 3D rotation) in the image formation process. Feature-based techniques and template matching are widely used to detect facial features. Available facial feature or face detection algorithms illustrate a good performance when illumination, scale and head orientation are controlled. However, tilt, rotation, scaling and unknown poses can cause several major problems for these approaches to work properly. Several methods are reported which demonstrated some ability to reduce these sensitivities. An alternative is to generate a 3D model of the face and accomplish the recognition in any given pose. In this study, following the latter approach, we develop an algorithm to model a specific human face in 3D so that it is possible to use the model in different applications such as computer animation for games or educational purposes and identification.

1.2 Problem Definition and Assumptions

3D face modeling is a special case of the general problem of 3D object representation. A 3D head model should include geometric and non-geometric properties of the object. Geometric properties of the faces may be obtained from the images taken from different viewpoints. This approach brings with itself the problem of determining location of a point in 3D space using the information available in the 2D images. This process which is referred to as 3D reconstruction needs

accurate knowledge about the physical properties of the camera by which the images are captured. Obtaining the physical properties of the camera is called camera calibration. In the absence of any knowledge related to camera parameters, we refer to the images as uncalibrated images. Incorporating properties of any specific camera in the 3D modeling algorithm restricts the method to a specific set of camera. Determining the physical properties of the camera by the user is also prone to error and it is a complex process for an ordinary user. Considering the problems in determining physical properties of the camera, we add this part as an integrated part of the algorithm.

The main aim of this thesis is generating 3D face models from uncalibrated images. Basic steps and the flow of the system are given in Figure 1.1. In the first part, the camera auto calibration is accomplished. This part starts with matching points from different views. In the matching step, a set of feature points provided by the user, is used to reduce erroneous correspondence. Using the physical parameters of the camera, a set of points in the form of a 3D cloud of random points is generated in the second part. These points are used in deforming the generic mesh. Final step is adding texture values to the model. Some restrictions on the image acquisition process and facial feature extraction are imposed. The restriction considered in our algorithm are listed below.

- All images are taken from the same elevation as the face and no tilt to left, right, forward or backward is present in the face position.
- No occluding object like eyeglasses, scarf etc. should cover the face totally or partially.
- Illumination should be consistent in all images.
- Hair should not cover the forehead.
- Face should have a neutral expression and no movement should exist form one image to the other.

The segmentation of the face and locating facial features are not considered as an integrated part of this thesis. The user provides the location of some feature points in the images manually. The location of the facial features given by the user are refined further to increase accuracy of the generated model. Neutral expression assumption is added here because our basic aim is generating a neutral model which can be later used in animation and exhibiting expressions. In addition, the rigid object and scene assumption is necessary in all camera calibration algorithms. Hence we assume that the face experiences no movement during image acquisition.

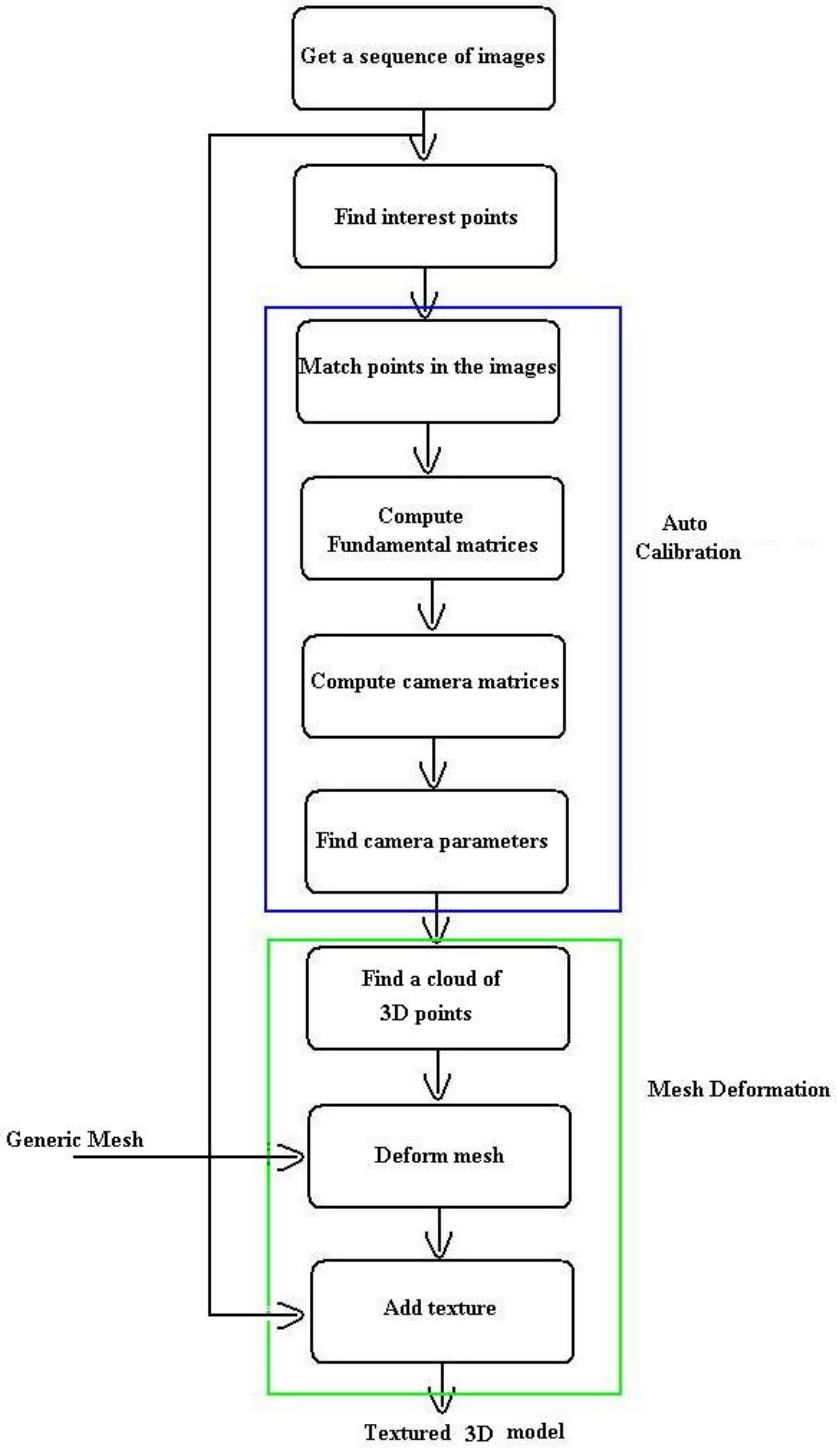

Figure 1.1: Basic steps and the flow of the system.

1.3 Contribution and Organization of the Thesis

The main aim of this thesis is generating 3D head models from uncalibrated images. The main contributions of the thesis can be stated as follows.

- A solution for the degeneracy in auto calibration when the camera rotation is about a single axis or parallel axes.
- Experimental and comparative study of the sensitivity of reconstruction and auto calibration algorithms to improper assumptions in intrinsic camera parameters.
- Devising an algorithm for mesh deformation using a cloud of 3D points and local refinements.
- Applying Delaunay triangulation method to local enhancements of human face model.
- Devising an algorithm for solving degeneracy in applying Delaunay triangulation to 3D meshes.
- Developing an integrated system for modeling human faces from uncalibrated images.

The details of methods and algorithms are discussed in 6 chapters as follows. Chapters 1 and 2 discuss projective and epipolar geometries and camera auto calibration. Chapter 3 provides the details of our solution for degeneracy in auto calibration and sensitivity of auto calibration to inaccurate assumptions in intrinsic camera parameters. The content of this chapter correspond to the auto calibration part of the system diagram in Figure 1.1. Chapter 4 presents face modeling basics and Chapter 5 includes the details of our mesh deformation algorithm. This chapter presents the details of mesh deformation part of the system diagram. Chapter 6 illustrates the experimental results and finally Chapter 7 concludes the thesis.

CHAPTER 2

Projective and Epipolar Geometries and Camera Model

In this chapter, we first introduce basic definitions of projective geometry which defines the image formation process and describes the necessary transformations for retrieving scene structure. Then, we proceed with epipolar geometry which elaborates on the relationships between images taken form the same scene at different view points. Another effective factor in relating images and reconstructing the scene is the physical properties of the camera. This chapter also presents the simple camera model and describes the relationships between camera properties and the images taken from a scene using projective and epipolar geometries. Gradual retrieving of camera parameters are presented in a stratified form. Furthermore, the basic properties of conics and quadrics are described. These properties play a significant role in determining camera parameters. The background information given in this chapter is used as a basis for the rest of the manuscript.

2.1 Introduction

Geometries are defined by their objects of study. Euclidean geometry is naturally preoccupied with distance and angle. Projective geometry studies properties unchanged by projections. Under a projection from one plane to another in 2D, a straight line stays a straight line. However, the distance between points or angle between two lines can vary. Even more disconcertingly, parallel lines may project to lines which meet. A good example is a straight railway track. The image formation process is a projective mapping form a 3D space to a 2D plane. A camera maps the real world points to image points. This projection can be modelled mathematically in homogeneous coordinate systems. However, the modeling depends on the specifications of the camera. These specifications are divided into two as intrinsic and extrinsic parameters. Intrinsic parameters define the physical properties of the camera. Extrinsic parameters specify the location and orientation of the camera. Images taken with different extrinsic camera parameters are related to each other. The relationship between multiple views of a scene from different view points is given by the epipolar geometry. The material presented in this chapter is based on the studies of Faugeras[1], Hartley[2].

2.2 Basic Definitions in Projective Geometry

The definitions of basic concepts in projective geometry are given below.

A *point* in projective n-space $\mathbb{P}^n$ is given by an $(n+1)$-vector of coordinates. These coordinates are called *homogeneous* coordinates. At least one of these coordinates should be different from zero. Given any two points $\mathbf{x}_1$ and $\mathbf{x}_2$, if there exist a nonzero scalar λ so that $\mathbf{x}_1 = \lambda\mathbf{x}_2$ then the two points are equal to each other. If the last coordinate of a point is zero, then the points is said to be *at infinity*.

A projective *line* may be viewed as a projective space of dimension one ($\mathbb{P}^1$).

A line in a projective plane is the set of points $[x, y, z]$ satisfying a homogenous linear equation $ax + by + cz = 0$. If the last coordinate of a point is zero, it lies on the *line at infinity*. A point on the line at infinity is called a *vanishing point*.

A projective *plane* is defined using $\mathbb{P}^2$. A point in this space is represented by its three coordinates $[x, y, z]$. The projective plane can be defined as an extension of the the Euclidean plane. If $\mathbb{E}$ is the set of points in $\mathbb{P}^2$ with non-zero last coordinates, then dividing by the last coordinate, any point can be written uniquely as $[x, y, 1]$. The point obtained this way can be identified with the point (x, y) in Euclidean plane. In a projective plane, every two distinct lines intersect at a unique point. The locus of points at infinity is the *plane at infinity* which is denoted by π_∞. The plane at infinity has the following properties.

- Two planes are parallel if and only if their line of intersection is on π_∞.
- A line is parallel to another line or a plane if the point of intersection is on π_∞.

The concepts defined for projective spaces $\mathbb{P}^1$ and $\mathbb{P}^2$ have analogies in projective space $\mathbb{P}^3$. A point $\mathbf{x}$ in 3-space is defined by the homogenous coordinates $\mathbf{x} = [x, y, z, w]$. A point in $\mathbb{P}^3$ with $w = 0$ lie on π_∞. A *transformation* is a general non-singular linear transformation of homogenous coordinates. A projective transformation projects every geometric object into a projectively equivalent object leaving all of its projective properties invariant. The class of invariants under each transformation is the basis for their classification. This classification and the invariants under each class of transformations are discussed in Section 2.4.

A *collineation* is a mapping that preserves *collinearity* which means if some points are on a straight line, they are mapped to a straight line. If the mapping is from an n-space into an m-space then the collineation is given by an $(n+1) \times (m+1)$ matrix $\mathbf{H}$.

A *homography* is a transformation in the projective plane which is denoted by a mapping $\mathbb{P}^2 \rightarrow \mathbb{P}^2$. A homography is an invertible mapping from points in $\mathbb{P}^2$ to points in $\mathbb{P}^2$ such that a group of points $\mathbf{x}_i$ $i = 1, \ldots, N$ lie on a line if and only if their transforms do.

2.3 Conics and Quadrics

A *conic* $\mathbf{C}$ is a curve in $\mathbb{P}^2$ defined using a second order equation. In the homogenous coordinates, a conic is given by $ax^2 + bxy + cy^2 + dx + ey + f = 0$. The equation can be written in matrix form as

$$\mathbf{x}^T\mathbf{C}\mathbf{x} = 0 \tag{2.1}$$

where $\mathbf{C}$ is given as

$$\mathbf{C} = \begin{bmatrix} a & b/2 & d/2 \\ b/2 & c & e/2 \\ d/2 & e/2 & f \end{bmatrix}$$

Using the above definition a conic in $\mathbb{P}^2$ is defined as the locus (the set of all points usually forming a curve or surface, satisfying some condition) of all points $\mathbf{x}$ satisfying Equation 2.1. Multiplying righthand side of Equation 2.1 by a scalar does not affect the equation. This means that a conic is defined only up to some scale and therefore it has five degrees of freedom. A *degenerate* conic has a defining matrix which is not of full rank.

A *dual conic* is the locus of all lines $\mathbf{l}$ tangent to the conic $\mathbf{C}$. The equation defining a dual conic can be given as $\mathbf{l}^T\mathbf{C}^*\mathbf{l}$. $\mathbf{C}^*$ is the adjoint of $\mathbf{C}$ which in real matrices is equal to $\mathbf{C}^{-1}$.

A *quadric* $\mathbf{Q}$ in the locus of points $\mathbf{X}$ in $\mathbb{P}^3$ satisfying the homogenous quadratic equation

$$\mathbf{X}^T\mathbf{Q}\mathbf{X} = 0 \tag{2.2}$$

where $\mathbf{Q}$ is a 4×4 symmetric matrix defined up to some scale factor. Similarly *dual quadric* $\mathbf{Q}^*$ is the locus of planes π satisfying the following homogenous

equation.

$$\pi^T \mathbf{Q}^* \pi = 0 \tag{2.3}$$

If $\mathbf{Q}$ is real then $\mathbf{Q}^* = \mathbf{Q}^{-1}$.

Absolute conic is a degenerate conic on π_∞ denoted by ω_∞. The points on ω_∞ satisfy

$$\begin{cases} x_1^2 + x_2^2 + x_3^2 = 0 \\ x_4 = 0 \end{cases} \tag{2.4}$$

$x_4 = 0$ indicates that all points belonging to the absolute conic should be on π_∞. For the points on π_∞ we have

$$\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \mathbf{I} \begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix}^T = 0$$

which means $\omega_\infty = \mathbf{I}$ and a conic of pure imaginary points. On π_∞, ω_∞ can be interpreted as a circle of radius $i = \sqrt{-1}$.

Dual of the absolute conic ω_∞ is a degenerate dual quadric in $\mathbb{P}^3$ called *absolute dual quadric* and denoted by Ω_∞^*. Any plane tangent to Ω_∞^* is also tangent to ω_∞. Ω_∞^* is represented by a 4×4 homogenous matrix of rank 3.

2.4 Stratification of Transformations

Each transformation in projective geometry leaves some properties of geometric objects invariant. An invariant is a property whose value is unchanged by a particular transformation. The transformations can be classified into three groups based on their invariants. The most general group in this classification belongs to projective transformations. This group has the least number of invariants. Second group is the group of affine transformations. This group may be considered as a restricted form of projective class. The last group is the group of Euclidean (metric) transformations and a subclass of affine class. The hierarchical relationship between the transformation classes has created the illusion of a stratification and hence the name stratum is used for each class. The

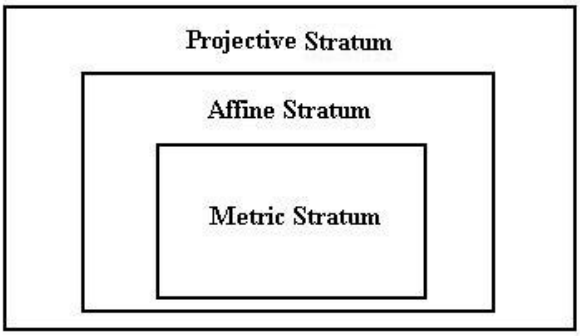

Figure 2.1: Stratification of the transformations.

relationship between the transformation classes is illustrated in Figure 2.1. The stratification and invariants in each stratum is discussed below.

2.4.1 Projective stratum

The projective stratum is defined by the class of projective transformations. These transformations are general non-singular linear transformations of homogeneous coordinates. In 3D space, the transformation is given by a 4×4 matrix which is defined up to some non-zero scale factor and has 15 independent elements. Invariants under a projective transformation are collinearity, tangency, order of contact and *cross-ratio*. Collinearity states that if a set of points lie on a line, they also remain on a line after transformation. Tangency refers to the fact that if a line is tangent to a curve, this state is preserved after transformation. If a line contacts a curve in several points, the transformation preserves order of these contacts.This is referred to as order fo contact. Cross ratio is defined as follows. Assuming that four points $\mathbf{x}_1$,$\mathbf{x}_2$,$\mathbf{x}_3$ and $\mathbf{x}_4$ are given, cross ratio of these points is defined as

$$Cross(\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \mathbf{x}_4) = \frac{|\mathbf{x}_1\mathbf{x}_2||\mathbf{x}_3\mathbf{x}_4|}{|\mathbf{x}_1\mathbf{x}_3||\mathbf{x}_2\mathbf{x}_4|} \tag{2.5}$$

where

$$|\mathbf{x}_i\mathbf{x}_j| = \det \begin{bmatrix} \mathbf{x}_{i1} & \mathbf{x}_{j1} \\ \mathbf{x}_{i2} & \mathbf{x}_{j1} \end{bmatrix}$$

The values of the cross ratio is independent from the homogenous representation chosen.

2.4.2 Affine stratum

The main difference between a projective transformation and an affine transformation lies in the concept of the plane at infinity π_∞. This plane is not part of the affine space and the points contained in π_∞ can not be expressed through the non-homogenous 3-vector coordinates. An affine transformation is usually written as a non-singular linear transformation followed by a translation. In 3D space, affine transformation is given as follows.

$$\begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} a_{14} \\ a_{24} \\ a_{34} \end{bmatrix} \tag{2.6}$$

In homogeneous coordinates this transformation is given by

$$\mathbf{T_A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.7}$$

In addition to all invariants under projective transformation an affine transformation includes three more invariants.

- **Parallel lines.** Parallel lines intersect at a point on the plane at infinity. In an affine transformation, this point is mapped to some other point on π_∞ preserving parallelism.

- **Ratio of lengths of parallel line segments.** Since affine transformations preserve parallelism, scaling magnitudes remain invariant in parallel lines.

- **Ratio of areas.** Variations in the areas depends on the product of scaling values in two directions. This factor is cancelled out when taking ratios.

2.4.3 Metric stratum

Metric stratum corresponds to the group of transformations which are a combination of orthogonal transformations and translations followed by a uniform scaling. The transformation matrix can be written as

$$\begin{bmatrix} s\mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.8}$$

where s is the scale factor, $\mathbf{R}$ is a 3×3 rotation matrix and it has the following properties.

$$\mathbf{R}\mathbf{R}^T = \mathbf{R}^T\mathbf{R} = \mathbf{I} \text{ and } \mathbf{R}^{-1} = \mathbf{R}^T$$

$\mathbf{t}$ is the 3×1 translation vector. The invariants under metric transformations are ratio of lengths and angles beside to all invariants under affine and projective transformations.

2.5 Camera Model

Before discussing how 3D information can be obtained from images, we show how images are formed. First, the pinhole camera model is introduced and then some important relationships between multiple views of a scene are presented. A digital camera samples light intensity over a physical area and produces an array of numbers. The physical area is discretised into a two dimensional (2D) rectangular array which is mapped onto the output array. The most common digital cameras are charge-coupled devices referred to as CCD cameras. Light

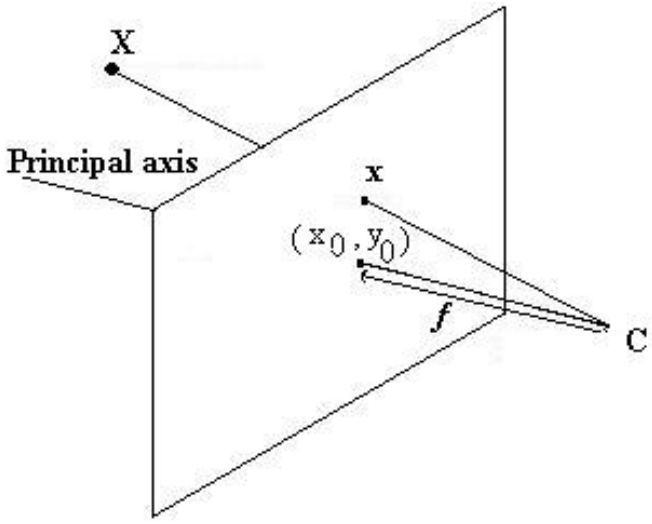

Figure 2.2: Image formation in a simple camera.

photons incident on the semiconductor light sensors generate a series of charges on the CCD array. These charges are transferred to an output register either directly one line at a time or the whole frame via a temporary storage area. In any case, the image formation process in a camera can be viewed as a mapping from the three dimensional (3D) world onto a 2D image. In the simplest form where the projection center is placed at the origin of the world frame and the image plane is at distance Z=f from this center, the projection process of a 3D point $\mathbf{X}$ to 2D point $\mathbf{x}$ can be modelled as follows.

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.9}$$

3D world points in homogeneous coordinates are related to image points via Equation 2.9. The projection process in this simple camera model is illustrated in Figure 2.2.

A pinhole camera is the simplest form of a projective camera which is modelled by its optical center $\mathbf{C}$ and its image plane. A 3D point $\mathbf{X}$ is projected into

an image point $\mathbf{x}$ given by the intersection of image plane with the line containing $\mathbf{C}$ and $\mathbf{X}$. The line containing $\mathbf{C}$ and orthogonal to the image plane is called the *principal axis* and its intersection with the image plane is the *principal point* (x_0, y_0). The distance between $\mathbf{C}$ and the image plane is the *focal length f*. In Equation 2.9, it is assumed that the principal point lies at the origin of coordinates in the image plane. In general, this may not be true. Furthermore, in Equation 2.9 it is assumed that the image coordinates are Euclidean coordinates having equal scales in both directions. In a CCD camera, there is the additional possibility of having non-square pixels. If the image coordinates are measured in pixels, this has the extra effect of unequal scale factors. Considering these features, a projective camera is defined using five parameters. These parameters are

- focal length f which defines the distance between the camera center to image plane,
- ratio of pixel width to pixel height λ which is called the *aspect ratio*,
- *skew* value γ which counts for the amount of deviation from a rectangular pixel,
- the location of principal point (x_0, y_0) on the image plane.

The transformation into the image coordinates is given by a matrix containing all camera properties. This matrix is called *intrinsic camera matrix* denoted by $\mathbf{K}$ and the parameters are referred to as *intrinsic camera parameters*. The intrinsic camera matrix is given in Equation 2.10.

$$\mathbf{K} = \begin{bmatrix} \alpha_x & \gamma & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.10}$$

where α_x and α_y are the focal length in terms of pixel width and pixel height respectively. Note that $\lambda = \frac{\alpha_x}{\alpha_y}$ is called aspect ratio. The focal length f is

implicitly given by α_x and α_y. Motion of the scene points from one position to another one also affects the image formation process. This motion can be modelled as follows.

$$\begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \tag{2.11}$$

where $\mathbf{R}$ is a 3×3 matrix showing the camera rotation and $\mathbf{t}$ a vector defining the translation of camera center. This relation defines a conversion between two Euclidean coordinate frames which are the world coordinate frame and the image coordinate frame. The translation and rotation of the camera are called *extrinsic camera parameters*. Combining intrinsic and extrinsic camera matrices with the relation given in Equation 2.9 which defines the mapping of the world points into image plane we get

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & \gamma & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{R} & \mathbf{t} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.12}$$

which can be written as

$$\mathbf{x} = \mathbf{KR}\,[\mathbf{I}|-\mathbf{t}]\,\mathbf{X} \tag{2.13}$$

or

$$\mathbf{x} = \mathbf{PX} \tag{2.14}$$

Note that in Equation 2.13 the notation $[\mathbf{I}|-\mathbf{t}]$ is used to show the concatenation of a 3×3 matrix $\mathbf{I}$ with a 3×1 vector $\mathbf{t}$ which gives a 3×4 matrix. The 3×4 matrix $\mathbf{P}$ is called *camera projection matrix*.

2.6 Epipolar Geometry

Images of a scene taken from different views are not completely independent [14]. Given a point $\mathbf{x}$ in the first image and camera parameters, the location of

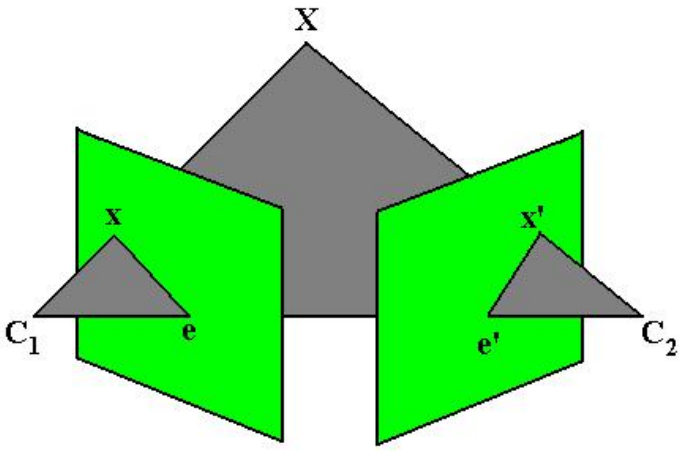

Figure 2.3: Epipolar plane and viewing cameras.

the point in the second image can be found. The reverse order is also possible. Given the images of a world point $\mathbf{X}$ in two images as $\mathbf{x}_1$ and $\mathbf{x}_2$, and the camera parameters, the world point $\mathbf{X}$ may be found. Even if the images are taken with different cameras having dissimilar internal parameters, a transformation relating the images to each other exists. Given sufficiently many matching point pairs, the relation from the first image to the second one can be determined. To show the nature of this transformation, consider two cameras $\mathbf{C}_1$ and $\mathbf{C}_2$ and a world point $\mathbf{X}$ which is mapped as $\mathbf{x}_1$ and $\mathbf{x}_2$ on the image plane of the cameras, respectively. The configurations are shown in Figure 2.3.

As it is clear form Figure 2.3, world point $\mathbf{X}$, image points $\mathbf{x}_1$ and $\mathbf{x}_2$ and camera centers $\mathbf{C}_1$ and $\mathbf{C}_2$ are coplanar. This plane is called *epipolar plane* and denoted here with π_e. The line connecting the camera centers is called *baseline* and the image of each camera center on the image plane of the other camera is called *epipole* and denoted by $\mathbf{e}$ and $\mathbf{e'}$ respectively. Epipoles are also intersection points of the baseline with image planes. Given an image point $\mathbf{x}$ and the camera centers, the other image point lies on the intersection of π_e with second image plane. This intersection gives a line which is known as *epipolar*

line.

2.6.1 Fundamental and Essential Matrices

Epipolar geometry can be represented in a mathematical form using a transformation which maps any point $\mathbf{x}$ on the first image to the epipolar line $\mathbf{l}_2$ in the second image [21]. Any point matching the point $\mathbf{x}$ should lie on this line. This mapping is represented by a matrix called *fundamental matrix*. The fundamental matrix $\mathbf{F}$ is a 3×3 of rank 2 homogenous matrix which satisfies

$$\mathbf{x}^T\mathbf{F}\mathbf{x} = 0 \tag{2.15}$$

for all corresponding points $\mathbf{x}_1$ and $\mathbf{x}_2$. Most important properties of the fundamental matrix can be expressed as follows.

- If $\mathbf{F}$ is the fundamental matrix between a pair of images $\mathbf{I}_1$ and $\mathbf{I}_2$, then $\mathbf{F}^T$ is the fundamental matrix between $\mathbf{I}_2$ and $\mathbf{I}_1$.
- $\mathbf{F}\mathbf{x}_1$ gives the epipolar line corresponding to point $\mathbf{x}_1$ in the second image and $\mathbf{F}^T\mathbf{x}_2$ is the epipolar line in the first image.
- All epipolar lines contain the epipole point. Thus $\mathbf{e}'^T(\mathbf{F}\mathbf{x}) = (\mathbf{e}'^T\mathbf{F})\mathbf{x} = 0$ and $\mathbf{e}'^T\mathbf{F} = 0$ which means that epipoles are right and left null spaces of the fundamental matrix.
- Since fundamental matrix is a 3×3 homogenous matrix and its determinant is zero, that is $Det\mathbf{F} = 0$, $\mathbf{F}$ has 7 degrees of freedom.
- Since $Det\mathbf{F} = 0$, one of its eigenvalues is zero and the other two are not equal in general.

In addition, it is important to note that the camera matrices depend on the image coordinate system and the world coordinate system. The reason lies in the role of camera matrices in relating 3D world measurements to image measurements. However, $\mathbf{F}$ does not depend on the choice of the world coordinate

system and it remains unchanged under any projective transformation.

Mapping world points to image points is directly affected by the internal camera parameters. This effect can be removed by normalizing coordinates when the intrinsic matrix $\mathbf{K}$ is known. If for each image point $\mathbf{x}$, we apply inverse of the intrinsic camera matrix, we obtain $\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{x}$ where $\hat{\mathbf{x}}$ is the normalized image point. Equation 2.14 is now given as $\hat{\mathbf{x}} = [\mathbf{R}|\mathbf{t}]\mathbf{X}$. The fundamental matrix corresponding to the normalized camera matrices is called *essential matrix*. Essential matrix $\mathbf{E}$ may be considered as the calibrated form of the fundamental matrix $\mathbf{F}$. The relationship between essential matrix and fundamental matrix is given by

$$\mathbf{E} = \mathbf{K}^T\mathbf{F}\mathbf{K} \tag{2.16}$$

where $\mathbf{K}$ is the camera intrinsic matrix. Essential matrix is a 3×3 rank 2 matrix with equal singular values [2]. An essential matrix is the product of a 3×3 orthogonal matrix, $\mathbf{U}$, and a 3×3 skew-symmetric matrix, $\mathbf{S}$.

$$\mathbf{E} = \mathbf{U}\mathbf{S}$$

In 3D Euclidean space, a translation and rotaion transformation is associated with an essential matrix.

2.6.2 Computing Fundamental Matrix

Equation 2.15 gives the definition of the fundamental matrix. Given a set of matching points we may obtain $\mathbf{F}$ using Equation 2.15. For two points $(x, y, 1)$ and $(x', y', 1)$ from Equation 2.15, we have:

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0$$

where f_{ij} are elements of $\mathbf{F}$. If we consider $\boldsymbol{f}$ as a vector made of elements of $\mathbf{F}$ in row order, then for a set of n points we will have:

$$\mathbf{A}\boldsymbol{f} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ x'_2x_2 & x'_2y_2 & x'_2 & y'_2x_2 & y'_2y_2 & y'_2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \boldsymbol{f} = 0$$

This equation shows a homogeneous equation which means $\mathbf{A}$ has at most rank 8. However, due to noise in finding points, it may be of rank 9. It means that the measurements are not exact and therefore, a least squares solution should be sought in solving this equation. The least square solution for $\boldsymbol{f}$ is the singular vector corresponding to the smallest singular value of $\mathbf{A}$. This vector is computed by decomposing $\mathbf{A}$ using $SVD(\mathbf{A}) = \mathbf{UDV}^T$ where $\mathbf{D}$ is a diagonal 9×9 matrix and $\mathbf{U}$ and $\mathbf{V}$ are orthogonal 9×9 matrices. This decomposition finds $\boldsymbol{f}$ subject to $\|f\| = 1$. Fundamental matrix $\mathbf{F}$ is a 3×3 matrix of rank 2 with $Det(\mathbf{F}) = 0$, but the solution we get from $SVD(\mathbf{A})$ is not of rank 2. To impose this constraint again singular value decomposition method is used. This time $SVD(\mathbf{F}) = \mathbf{UDV}^T$ is found. $\mathbf{D}$ is a diagonal matrix as $diag(r, s, t)$ satisfying $r >= s >= t$ where $diag(r, s, t)$ indicates a 3×3 diagonal matrix having r, s and t as its diagonal elements. $\mathbf{F}'$ found by assigning zero to t in matrix $\mathbf{D}$ satisfies the constraint.

$$\mathbf{F}' = \mathbf{U}\ diag(r, s, 0)\ \mathbf{V}^T$$

This method is called 8 points algorithm [10]. The important matter in 8 point algorithm is proper normalization of the input data which leads to a large improvement in the solution and stability of the result. The normalization is a translation and scaling of each image points so that the centroid of the points is at the origin and the root mean squares (RMS) distance of the points from the origin is $\sqrt{2}$. The second point in this algorithm is that generally we get a large number of point pairs from interest point detection and matching. Some of these

points may have errors in computing the locations and in yet some others there may be false matches. To find the best approximation for fundamental matrix different improvements have been proposed. The following algorithm which is a slightly modified form of RANSAC [2] algorithm is used here. Our algorithm modifies original RANSAC algorithm by using 8 point algorithm instead of 7 point algorithm. This modification prevents generation of more than a single solution for **F** since 7 point algorithm generates 3solutions while 8 point algorithm generates a unique solution. After determining **F** epipoles can be found.

Algorithm 1 Computing fundamental matrix

1 Choose 8 points from the matching point set randomly.

2 Find the Fundamental matrix **F**.

3 For each point, find its distance form epipolar line and group the points as inliers and outliers using a threshold on the computed distance.

4 Repeat steps 1 through 3 for n times.

5 Choose the output with maximum number of inliers as the most reliable output and recompute the fundamental matrix using inliers of this iteration.

Epipoles $\mathbf{e}$ and $\mathbf{e}'$ which are the images of camera centers in the image plane of the other camera are left and right null vectors of the fundamental matrix. Epipoles which are used in estimating camera matrices are defined as

$$\mathbf{Fe} = 0$$

$$\mathbf{e'F} = 0$$

and can be found by singular value decomposition of **F**. Epipoles are used in computing camera projection matrices.

CHAPTER 3

Approaches in Auto Calibration

The main goal of all 3D reconstruction algorithms is to obtain 3D coordinate values of any point in the scene from its images. The reconstruction however, is not always in the Euclidean frame which we naturally take for evident. Without any knowledge about the scene or the intrinsic camera parameters, we may reach only up to a projective reconstruction. In fact when the intrinsic camera parameters are not known, fundamental matrix $\mathbf{F}$ is the only available relationship between the image points. Any information about the scene or camera may be used to upgrade a projective reconstruction to a higher level. Upgrading a reconstruction using only *a-priori* partial knowledge about the intrinsic camera parameters is called *auto calibration*. Any knowledge about the camera intrinsic parameters can be used to upgrade a reconstruction from projective to metric. This knowledge can be in the form of known and/or unknown but constant values for camera parameters. Auto calibration methods may be divided into two basic groups. First group considers general motion of the camera while second group restricts camera movement to some specific forms such as planar movement, pure translation or pure rotation. Each group is reviewed separately in the following sections. The scene knowledge such as parallel or vertical edges of

the objects can be used in a stratified method for upgrading the reconstruction from projective to affine and metric. Furthermore, images of a known calibrated grid can be used for camera calibration. In this chapter, the major approaches in auto calibration are discussed in a comparative manner.

3.1 General Motion Algorithms

In a general motion auto calibration algorithm, it is assumed that the camera motion is unrestricted [7, 4, 19]. This corresponds to a camera which freely moves around. An example for a general motion is a hand-held camera. This means that no *a-priori* knowledge about the camera position, translation or rotation is assumed in the algorithm. However, some special motions may result in degenerate solutions. Degeneracy and the motions causing it are discussed in Chapter 4. Here, we assume that the general motions do not cause any degeneracy.

3.1.1 Auto Calibration using Kruppa Equations

Kruppa equations are originally introduced by the work of Faugeras, Luong and Maybank [5]. These equations are considered to be the first solution to auto calibration problem. Kruppa equations are based on the epipolar restriction. They state that the epipolar lines corresponding to epipolar planes tangent to the absolute conic, should be tangent to its projection in both images. These epipolar planes which are passing through both camera centers $\mathbf{C}_1$ and $\mathbf{C}_2$ are tangent to ω_∞ on the plane at infinity π_∞. Kruppa equations imposes that the epipolar planes should also be tangent to the images of ω_∞ in the image planes of the cameras. This situation is illustrated in Figure 3.1. We assume that ω and ω' are the images of a world conic Ω and ω^* and $\omega^{*'}$ are their duals. Let $\mathbf{H}$ show the homography matrix from first image to the second one , we have [1]

$$[\mathbf{e}']_\times \omega^{*'} [\mathbf{e}']_\times = \mathbf{H}^{-T} [\mathbf{e}]_\times \omega^* [\mathbf{e}]_\times \mathbf{H}^{-1} \tag{3.1}$$

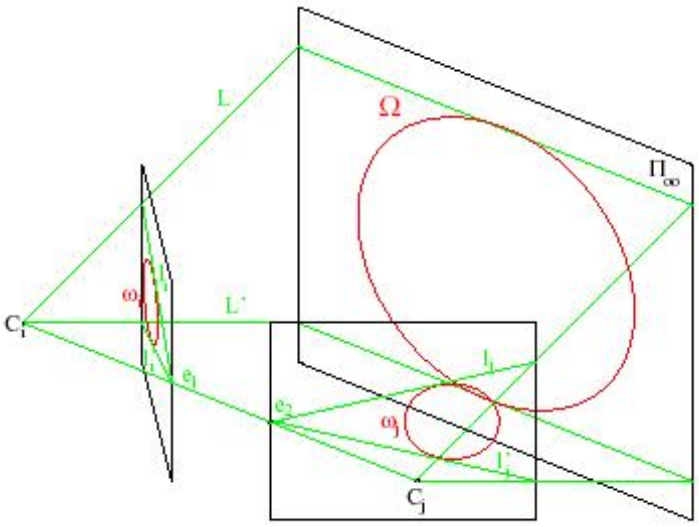

Figure 3.1: Epipolar constraint of Kruppa equations.

where $[\mathbf{e}]_\times$ is a skew-symmetric matrix defined as

$$[\mathbf{e}]_\times = \begin{bmatrix} 0 & -e_3 & e_2 \\ e_3 & 0 & -e_1 \\ -e_2 & e_1 & 0 \end{bmatrix} \tag{3.2}$$

Equation 3.1 is valid because $[\mathbf{e}']_\times \omega^{*'} [\mathbf{e}']_\times$ is a degenerate point conic. However, using Equation 2.6.2 the right hand side of Equation 3.1 is equal to $\mathbf{F}\omega^*\mathbf{F}^T$. On the other hand, from the properties defined for ω^*_∞ in Chapter 2, we know that ω^*_∞ is fixed under any similarity homography. Therefore, 3.1 is written as

$$[\mathbf{e}']_\times \omega^{*'}_\infty [\mathbf{e}']_\times = \mathbf{F}\omega^*_\infty \mathbf{F}^T \tag{3.3}$$

Considering the relationship between $\omega^{*'}_\infty$ and intrinsic camera parameters $\mathbf{K}$, auto calibration can be performed using Equation 3.3. This relation gives the basic idea behind Kruppa equations. It worths noting that no projective reconstruction is necessary in this method which means that computing camera projective matrices is not necessary. Assuming constant internal parameters in all views, which can be achieved if all images are taken using the same camera,

Kruppa equations provide two constraints on the intrinsic parameters. Given three images and finding $\mathbf{F}$ between each pair, six constraints are obtained which are sufficient to solve the equations. One disadvantage of Kruppa equations is that they provide no constraint when there is no translation between images. This is clear from the Equation 3.3 because in case of no translation we have $\mathbf{F} = [\mathbf{e}']_\times$. Another disadvantage of using the Kruppa equations is the complexity of the solutions [6]. However, when there are only one pair of images, Kruppa equations are the only constraint available.

3.1.2 Modulus Constraints

The plane at infinity can be used to find camera parameters. The property of the homographies of the plane at infinity are used for this purpose [15]. Infinity homography $\mathbf{H}_{ij}^{\infty}$ from view i to view j can be written as [15]:

$$\mathbf{H}_{ij}^{\infty} = \mathbf{K}\mathbf{R}_{ij}^{T}\mathbf{K}^{-1} \tag{3.4}$$

where $\mathbf{R}_{ij}$ is the rotation matrix that describes the relative orientation of the j^{th} camera with respect to the i^{th} one. From Equation 3.4 it is clear that $\mathbf{H}_{ij}^{\infty}$ is conjugated with a rotation matrix which implies that its eigenvalues have the same moduli. Writing the characteristic equation of $\mathbf{H}_{ij}^{\infty}$ we have:

$$Det(\mathbf{H}_{ij}^{\infty} - \lambda I) = l_3\lambda^3 + l_2\lambda^2 + l_1\lambda + l_0 \tag{3.5}$$

The necessary condition for the roots of Equation 3.5 to have the same moduli is given as

$$l_3 l_1^3 = l_2^3 l_0 \tag{3.6}$$

With three images, three pairs of constraints of the type given in the Equation 3.6 on the parameters of π_∞ are obtained. This yields a limited number of solutions. Some of these can be omitted considering only real valued cases. Having located the plane at infinity, image of dual absolute conic can be found

as follows.

$$\omega_{\infty}^{*} = \mathbf{H}_{\infty 1i}\mathbf{K}\mathbf{K}^{T}\mathbf{H}_{\infty 1i}^{T} \tag{3.7}$$

Camera parameters are found from Cholesky decomposition of ω_{∞}^{*}.

$$\omega_{\infty}^{*} = \mathbf{K}\mathbf{K}^{T} \tag{3.8}$$

3.1.3 Auto Calibration using Absolute Dual Quadric

Since the absolute conic is constant under Euclidean transforms, its images are also invariant under a moving camera if the intrinsic camera parameters are constant. This is the main idea used in some auto-calibration methods. Triggs [17] proposes to determine absolute dual quadric by assuming that images of the absolute conic are constant in all cameras. The problem here is that the relations are homogeneous and therefore the equalities satisfy only up to some unknown scale factor. Triggs solve this problem by considering the cross product of the ratio. This gives a set of quadratic equations which provides intrinsic camera parameters. The absolute dual quadric Ω_{∞}^{*} is a symmetric 4×4 degenerate matrix of rank 3. Ω_{∞}^{*} projects to ω_{∞}^{*} by the following equation.

$$\omega_{\infty}^{*} = \mathbf{P}\Omega_{\infty}^{*}\mathbf{P}^{T} \tag{3.9}$$

ω_{∞}^{*} is related to intrinsic camera parameters via Equation 3.8 which means that if intrinsic camera parameters are constant in all images, ω_{∞}^{*} is the same in all projections and the right hand side parts in Equation 3.9 are equal up to some unknown scale factor. Since Ω_{∞}^{*} is symmetric, there are only 10 unknowns and given at least three images taken with constant intrinsic parameters, they may be determined. The method is easily extendable to cases where only some of the intrinsic parameters are constant or some of them are known. This feature is very important when the images are taken with the same camera with different focus settings. To solve the scale factor problem Triggs proposes using cross product to eliminate scale factor as follows.

$$\|\omega_{\infty}^{*}(ab) \wedge (\mathbf{P}_i\Omega_{\infty}^{*}\mathbf{P}_i^{T})(cd)\|^2 = 0 \tag{3.10}$$

where ab and cd refer to an element at row a, column b, and row c, column d, respectively. $\wedge$ denotes cross product so $A(12) \wedge B(34) = A(12) \times B(32) - A(34) \times B(12)$. Triggs minimizes $15m$ equations of this type arising from bilinear quadratic projection constraints given by Equation 3.10 subject to $Det(\Omega^*_\infty) = 0$. Here m is the number of views. Heyden and Äström [41] propose a similar method but they consider scale factors as unknown parameters and assume that the first camera matrix is $\mathbf{P}_1 = [\mathbf{I}_{3\times3}|\mathbf{0}]$. If the position of the principal point is known, the method can be simplified to a set of linear equations. If we assume the principal point to be at the origin of the coordinate system, then Ω^*_∞ is defined is terms of intrinsic camera parameters as follows.

$$\Omega^*_\infty = \begin{bmatrix} \alpha_x^2 + \gamma^2 & \gamma\alpha_y & 0 \\ \gamma\alpha_y & \alpha_y^2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.11}$$

This equation turns out to be a set of linear equations. However, since the image measurements are subject to noise and inaccuracy in matching points in the images, a least squares solution is preferred which can be found using singular value decomposition.

3.1.4 Auto Calibration using Essential Matrix

Essential matrix is the calibrated form of the fundamental matrix which is related to fundamental matrix by Equation 2.16. Since fundamental matrix $\mathbf{F}$ has rank 2, one of the eigenvalues is zero. $\mathbf{E}$ also is of rank 2 but the non-zero eigenvalues are equal. This puts a constraint for computing the camera intrinsic parameters from $\mathbf{F}$. Auto calibration by means of essential matrix is used by Hartley [31], Faugeras [26], Luong [25] to find $\mathbf{K}$. The method is first introduced by Hartley where he uses only two images. He shows that using two images, it is only possible to extract at most two parameters of the intrinsic camera matrix. The method is later extended to a larger image sequence by Mendonça and Cipolla[35] which makes possible to extract all camera parameters and to

increase stability of the algorithm. Assuming similar internal parameters in all cameras, a cost function can be defined as follows.

$$C(K) = \sum_{i=1}^{n} \sum_{j=i}^{n} w_{ij} \frac{1_{\sigma ij} - 2_{\sigma ij}}{1_{\sigma ij} + 2_{\sigma ij}} \tag{3.12}$$

where w_{ij} is a coefficient showing the degree of confidence in the computation of fundamental matrix and $1_{\sigma ij}$ and $2_{\sigma ij}$ are two non-zero singular values of $\mathbf{E}$ in descending order between images i and j. Assuming all internal parameters fixed but unknown means that we should have at least three different images taken from different views. This fact has been pointed by Hartley [2] as:

$$n \times n_k + (n-1) \times n_f = 8 \tag{3.13}$$

where n is the number of views, n_k the number of known parameters and n_f number of fixed but unknown parameters. Mendonça and Cipolla in their algorithm have considered a large sequence of images to increase stability and robustness but assumed $\lambda = 0$. A slightly different implementation of this method and experimental results are also given by Fusiello [27].

3.2 Restricted Motion Algorithms

Any knowledge about the type of camera motion can be incorporated into the calibration algorithms. Among camera motions some restricted motions are of interest. Restricted motions can make camera calibration algorithms simpler. However, in some of these motions, all camera parameters can not be found. Yet, in some other group of restricted motions, camera calibration is possible but scene reconstruction can not be done. In addition, the motions in real world applications are not exactly restricted motions, but near to them which means a new source of inaccuracy is being added to the algorithms. However, the simplicity and speed of computation makes these algorithms attractive for the applications where high accuracy is not required. Furthermore, the results obtained from these algorithms may be used as initial values for other methods.

3.2.1 Pure translation

Van Gool [37] shows that when the camera moves only in a pure translation form, the camera matrices become $\mathbf{P}_1 = [\mathbf{I}_{3\times3}|\mathbf{0}]$ and $\mathbf{P}_2 = [\mathbf{I}_{3\times3}|-\mathbf{t}]$. He also shows that $\mathbf{P}_1 = [\mathbf{I}_{3\times3}|\mathbf{0}]$ and $\mathbf{P}_2 = [\mathbf{I}_{3\times3}|\mathbf{e}]$ give an affine reconstruction. The latter result is easily obtained by noting that these camera matrices are found from metric ones through an affine transformation. A world point $\mathbf{X}$ is projected to these images as:

$$\begin{aligned} s_1\mathbf{x}_1 &= \mathbf{K}[\mathbf{I}|\mathbf{0}]\mathbf{X} \\ s_2\mathbf{x}_2 &= \mathbf{K}[\mathbf{I}|-\mathbf{t}]\mathbf{X} \\ s_2\mathbf{x}_2 &= s_1\mathbf{x}_1 + s\mathbf{e} \end{aligned} \tag{3.14}$$

Considering the above equations, a homography from the first image to the second image is given by $\mathbf{H}_{12}^{\infty} = \mathbf{I}_{3\times3}$ and therefore, this method can not provide any constraint to find $\mathbf{K}$ and a metric reconstruction. Armstrong modified this method by adding new views with rotation. He first finds the affine reconstruction from pure translation and then upgrades to metric reconstruction by using additional views [24].

3.2.2 Pure rotation

In the case of pure rotation, the camera matrices become $\mathbf{P}_1 = [\mathbf{I}|\mathbf{0}]$ and $\mathbf{P}_i = [\mathbf{R}_i|\mathbf{0}]$. A world point $\mathbf{X} = [\mathbf{x}^T w]^T$ is projected to the first image as $\mathbf{x}_1 = \mathbf{K}\mathbf{x}$ and to image i as $\mathbf{x}_i = \mathbf{K}\mathbf{R}_i\mathbf{x}$. This gives the relationship between the image points with a homography such as $\mathbf{H}_{1i} = \mathbf{K}\mathbf{R}_i\mathbf{K}^{-1}$. This relation is valid for points at infinity and can be written as $\mathbf{x}_i^{\infty} = \mathbf{K}\mathbf{R}_i\mathbf{x}$. On the other hand form the definition of the dual of the absolute conic ω_{∞}^* we have:

$$\omega_{\infty}^* = \mathbf{H}_{ij}^{\infty}\omega_{\infty}^*\mathbf{H}_{ij}^{\infty T} \tag{3.15}$$

This relation can be used to find the camera intrinsic parameters $\mathbf{K}$. However, it should be noted that images of the world point $\mathbf{X}$ are independent from the last

coordinate component w. This means that despite possibility of auto calibration in this type restricted motion, 3D reconstruction is not possible.

3.2.3 Planar motion

Planar motion can be described as a translation in a plane and a rotation about an axis normal to this plane. If all motions in a sequence are planar and rotation axes are parallel, then it is called a planar motion sequence. The plane at infinity and the absolute conic are invariant in auto calibration methods. These two invariants are used in all auto calibration methods to find the camera parameters. In planar motions, two more invariants are added which are used in finding the plane at infinity. These points are the vanishing point of the rotation axis and the vanishing points of the motion plane. The intersection points of the vanishing line of the motion plane or horizon with the absolute conic are also fixed. To extract these points, it is necessary to find the fixed points. The horizon is the image of the plane through all centers of projections. The vanishing point is located as the intersection of the images of all rotation axes. Planar motion has been used for camera calibration by different groups such as Armstrong *et al* [24], Agapito *et al* [23], Oliensis [36] and Triggs [39].

CHAPTER 4

Auto Calibration with Single Axis Rotations and Small Translations

Auto calibration in the general motion case is subject to some difficulties arising from its intrinsic limitations [38]. In some specific classes of motions, camera parameters can not be recovered. These classes of motions and the related difficulties are independent from the auto calibration methods. In these cases which are referred to as *degenerate cases*, either no solution can be found for a camera parameter or the solution is not unique. One of these degenerate cases which frequently happens in human face modelling applications, is having rotations about parallel axes. This type of degenerate motion makes camera calibration and 3D reconstruction impossible. Small translation also deteriorates the reconstruction phase, because the intersection point of the image lines are far from the image plane and hence subject to computation error. This chapter gives an insight into the degeneracy problem and the solution adopted for face reconstruction application. The reliability of the solution is also verified experimentally.

4.1 Degenerate Motions

In Equation 3.8, we show that the intrinsic camera parameters are related to a degenerate point conic called absolute conic. The absolute conic is an imaginary conic lying on the plane at infinity and has a fixed projection in all images. The degeneracy problems occur when the absolute conic is not the only conic having a fixed projection in all images. In this case, there is no way to determine which conic corresponds to the real absolute conic and the motion sequence is called *critical motion* with respect to auto calibration. The term critical motion is used because the degeneracy caused in this way is independent from the internal camera configuration.

Any proper imaginary conic on π_∞ is given by a 3×3 symmetric matrix $\mathbf{C}$. The image of $\mathbf{C}$ in view i is given by $\mathbf{c}_i = \mathbf{R}_i \mathbf{C} \mathbf{R}_i^T$ where $\mathbf{R}$ is the rotation matrix. To have the identical images of a conic, we should have $\mathbf{c}_i \simeq \mathbf{c}_j$ where $\simeq$ stands for equality up to a scale factor. This means that we should have

$$\mathbf{R}_i \mathbf{C} \mathbf{R}_i^T = \mathbf{R}_j \mathbf{C} \mathbf{R}_j^T \tag{4.1}$$

where the $\simeq$ sign has changed to $=$ because $\mathbf{R}$ is a rotation matrix and therefore its determinant is equal to 1. From Equation 4.1 the critical motions can be found as [1]:

- a rotation of angle zero (pure translation);
- an arbitrary rotation around a fixed axis $\mathbf{r}$;
- a 180° rotation around an axis normal to a fixed direction $\mathbf{r}$.

In the first case, cameras are translated with respect to each other and all proper imaginary conics in π_∞ have same images in all views. In the other two cases, the conics are given as a family of parametric matrices

$$\mathbf{C} = \lambda \mathbf{I}_3 + \mu \mathbf{r} \mathbf{r}^T$$

where λ and μ are arbitrary parameters and we end up with several solutions. Considering $\mathbf{R} = \mathbf{R}_j^T \mathbf{R}_i$ as the relative rotation between views i and j we may have the following relation from Equation 4.1.

$$\mathbf{RC} = \mathbf{CR} \tag{4.2}$$

This equation imposes the restriction that $\mathbf{R}$ should leave the eigenvalues of $\mathbf{C}$ invariant. To satisfy this restriction, the potential absolute conic lying on π_∞ should belong to one of the following groups considering its eigenvalues.

1. Three equal eigenvalues.

2. Two equal and one distinct eigenvalues.

3. Three distinct eigenvalues.

Only the real absolute conic itself corresponds to the first case, which is therefore not a degenerate case. In the second case, the eigenspace corresponds to a plane and a line orthogonal to that plane. An arbitrary rotation around that line, or, a rotation of 180° around a line in the plane which is incident to the other line, leaves the image of the conic unchanged. In the third case, the eigenspace consists of three mutually orthogonal lines. The rotations of 180° around one of these lines leave the image of the conic unchanged. We may classify this group of critical motions as follows.

1. Motion sequences for which all rotations are by an arbitrary angle about an axis on a plane or by 180° about an axis normal to the plane.

2. Motion sequences for which all rotations are by 180° about mutually orthogonal axes.

3. Motion sequences for which all rotations are by 180° about an specific axis.

4. Motion sequences for which no rotation takes place at all.

The above groups of degenerate motions have practical importance, since some of the critical motion sequences are often used for acquiring images. One of the cases in practice is translating arbitrarily while rotating about a fixed direction. This is the case when we are taking images of human faces from different views. Next section describes our solution for this kind of degeneracy [12].

4.2 Removing Degeneracy

Auto-calibration methods discussed in Chapter 2 are applicable if there are rotations about at least two non-parallel axes. The ambiguity can be described as follows: since absolute conic lies on the plane at infinity, its images on the different views are related to each other by a planar infinite homography $\mathbf{H}_\infty^i$.

$$\omega_\infty^* = \mathbf{H}_\infty^i \omega_\infty^* \mathbf{H}_\infty^{iT} \tag{4.3}$$

However, $\mathbf{H}_\infty^i$ is related to internal and external parameters by Equation 4.4.

$$\mathbf{H}_\infty^i = \mathbf{K}\mathbf{R}_i\mathbf{K}^T \tag{4.4}$$

Assuming $d\mathbf{r}$ to be the unit eigenvector of R, we have $\mathbf{R}d\mathbf{r} = d\mathbf{r}$ and the eigenvector of $\mathbf{H}_\infty^i$ is $\mathbf{v} = \mathbf{K}d\mathbf{r}$. This means that a one-parameter family of solutions such as $\omega_\infty^*(\mu) = \omega_\infty^* + \mu\mathbf{v}\mathbf{v}^T$ satisfy Equation 4.3 and the solution is degenerate and not unique. To obtain a unique solution, two more restrictions are added which are: skew $\gamma = 0$ and aspect ratio $\lambda = 1$. These restrictions are reasonable assumptions in most real imaging systems. Assuming $\gamma = 0$ gives a unique solution when the rotation is about an axis which is not parallel to one of the axes of the camera. On the other hand, if the rotation is about an axis parallel to $\mathbf{x}$ axis, assuming $\gamma = 0$, ω_∞^* becomes:

$$\omega_\infty^* = \begin{bmatrix} \alpha_x^2(1+\mu) + x_0^2 & x_0 y_0 & x_0 \\ x_0 y_0 & \alpha_y^2(1+\mu) + y_0^2 & y_0 \\ x_0 & y_0 & 1 \end{bmatrix} \tag{4.5}$$

where x_0 and y_0 are camera principle point coordinates and α_x and α_y are the camera resolution in the unit of length in **x** and **y** directions respectively. In general, if the rotation axis is parallel to one of the camera axes, the solution is parametric in μ and the ambiguity becomes as given in Table 4.1

Table 4.1: The relation between ambiguity and rotation axis

rotation axis	ambiguity
x-axis	α_x can not be determined
y-axis	α_y can not be determined
z-axis	α_x and α_y can not be determined

Therefore in the first two cases, the second restriction, that is assuming $\lambda = 1$ leads to a unique solution. However, when the rotation is about **z** axis, since two values are parametric, a unique solution is not possible.

4.3 Sensitivity of Internal Camera Parameters

To evaluate the results obtained from any camera calibration algorithm and verify the effect of different assumptions in solving the calibration problem, an error evaluation method is described in this section. In this method, we assume that the intrinsic and extrinsic camera parameters are known for evaluation purposes. Random points from a unit sphere in front of the cameras are generated and their images are mapped by the cameras from different view points. The images are then used as input to the auto-calibration algorithms and their 3D values are recovered considering the global scale which is available and compared with their true values. The main advantage of this testing method is that we can change the intrinsic camera parameters freely and consider the reconstruction error caused by these modifications. Furthermore, it is possible to consider the effect of initialization or some degenerate values in the convergence of the algorithms. We also consider the effect of noise in the detection and matching the points by adding a zero mean Guassian noise with different standard deviation values to the position of points in the images. n different cameras are generated

by assuming the first camera at the center of the coordinate system and the others translated by $\mathbf{t}_i$ and rotated by $\mathbf{R}_i$. The camera matrices therefore are: $\mathbf{P}_1 = [\mathbf{I}_{3\times3}|\mathbf{0}]$ and $\mathbf{P}_i = \mathbf{K}[\mathbf{R}_i|\mathbf{t}_i]$ i=2,..,n. Images of a set of 3D points ($\mathbf{X}$) are generated from $\mathbf{x}_i = \mathbf{P}_i\mathbf{X}$ where the subscript i refers to the camera being used. The reconstruction of the points is done by finding intrinsic parameters $\mathbf{K}$ and homography $\mathbf{H}$ matrices and then computing 3D coordinates of the points by triangulation [2]. Since true values of coordinates of the points are known, overall scaling of the scene is straightforward. The method is used in:

1. evaluation of the auto-calibration method used to find intrinsic parameters,
2. evaluation of the effect of some assumptions such as vanishing skew, aspect ratio=1 or principal point at the origin,
3. reconstruction errors due to the assumptions in intrinsic parameters,
4. reconstruction errors due to the mismatches in finding point pairs.

First group of experiments which includes nine cases tests the sensitivity of each method to the assumptions concerning internal camera parameters. We consider assumptions on skew, aspect ratio, and location of the principal point. For each parameter, three different tests with different levels of error in the assumed parameter are conducted. Second group of experiments examines the sensitivity of each method to noise in image points. Gaussian noises with three different standard deviation values are examined. Each test case is carried out using 100 random points from a unit sphere in front of five viewing cameras. The results are drawn by repeating the test cases with different random points for 100 times and getting their average values. Tables 4.2 to 4.10 demonstrate the results of the first group of tests. If $X_i - X_i'$ is the difference between computed and true values, Root Mean Square of Reconstruction Error RE_{rms} is defined as:

$$RE_{rms} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X_i - X_i')^2}$$

Table 4.2: Effect of a wrong assumption in skew ($\gamma = 0$ assumed, true value $\gamma = 10$).

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	10	1	0	0	6.20	–	2.24	0.00	0.00	0.11
$Using\Omega^*_\infty$	800	10	1	256	256	2.30	–	0.31	3.18	2.12	0.02
$Using\mathbf{E}$	800	10	1	256	256	1.67	–	0.61	4.33	0.11	0.04

Table 4.3: Effect of a wrong assumption in skew ($\gamma = 0$ assumed, true value $\gamma = 30$).

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	30	1	0	0	13.20	–	8.3	0.00	0.00	1.02
$Using\Omega^*_\infty$	800	30	1	256	256	4.80	–	0.29	4.10	3.12	0.05
$Using\mathbf{E}$	800	30	1	256	256	6.90	–	0.70	5.40	5.86	0.14

Table 4.4: Effect of a wrong assumption in skew ($\gamma = 0$ assumed, true value $\gamma = 50$).

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	50	1	0	0	23.16	–	12.71	0.00	0.00	1.97
$Using\Omega^*_\infty$	800	50	1	256	256	13.10	–	7.30	3.20	3.15	1.54
$Using\mathbf{E}$	800	50	1	256	256	19.23	–	6.80	17.18	11.32	1.72

Table 4.5: Effect of a wrong assumption in aspect ratio ($\lambda = 1$ assumed and true value $\lambda = 1.05$).

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	0	1.05	0	0	7.01	26.20	–	6.10	7.30	0.60
$Using\Omega^*_\infty$	800	0	1.05	256	256	4.14	12.30	–	1.30	1.19	0.35
$Using\mathbf{E}$	800	0	1.05	256	256	6.08	33.00	–	1.10	2.30	0.43

Table 4.6: Effect of a wrong assumption in aspect ratio ($\lambda = 1$ assumed and true value $\lambda = 1.1$).

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	0	1.1	0	0	37.10	12.60	–	4.23	3.92	1.63
$Using\Omega^*_\infty$	800	0	1.1	256	256	12.03	11.40	–	3.11	1.12	0.91
$Using\mathbf{E}$	800	0	1.1	256	256	24.25	15.25	–	1.56	2.50	1.42

Table 4.7: Effect of a wrong assumption in aspect ratio ($\lambda = 1$ assumed and true value $\lambda = 1.3$).

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	0	1.3	0	0	89.4	9.10	–	19.29	11.68	2.87
$Using\Omega^*_\infty$	800	0	1.3	256	256	63.20	12.90	–	6.18	2.15	1.97
$Using\mathbf{E}$	800	0	1.3	256	256	95.10	8.10	–	0.58	0.21	2.54

Table 4.8: Effect of a wrong assumption in the location of principal point ($x_0 = 0$,$y_0 = 0$ assumed and true values $x_0 = 10$,$y_0 = 10$).

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	0	1	10	10	7.23	8.10	5.10	–	–	1.14
$Using\Omega^*_\infty$	800	0	1	10	10	1.10	0.90	0.23	–	–	0.71
$Using\mathbf{E}$	800	0	1	10	10	2.30	0.80	0.75	–	–	0.98

Table 4.9: Effect of a wrong assumption in the location of principal point ($x_0 = 0$,$y_0 = 0$ assumed and true values $x_0 = 25$,$y_0 = 25$).

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	0	1	25	25	16.1	17.8	15.30	–	–	2.12
$Using\Omega^*_\infty$	800	0	1	25	25	6.90	2.10	1.22	–	–	1.62
$Using\mathbf{E}$	800	0	1	25	25	9.10	3.90	0.96	–	–	1.93

Table 4.10: Effect of a wrong assumption in the location of principal point ($x_0 = 0$,$y_0 = 0$ assumed and true values $x_0 = 50$,$y_0 = 50$).

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	0	1	50	50	29.65	28.10	12.10	–	–	5.91
$Using\Omega^*_\infty$	800	0	1	50	50	16.40	0.70	5.30	–	–	2.29
$Using\mathbf{E}$	800	0	1	50	50	18.50	0.30	7.38	–	–	2.50

The results present the sensitivity of reconstruction error to wrong assumptions in each internal camera parameter. The results show that invalid assumptions in the skew parameter smaller error effects in the reconstruction values and in the estimation of the other parameters. Assumptions on the principal point location are generally less reliable. This is particularly evident from the results of the linear method. The results given above are the average of the values obtained in 100 tests however, the cases where the algorithm does not converge have not been considered. These cases happen when the noise effect or improper assumption on camera parameters makes data inconsistent for metric reconstruction. Second group of testing cases are related to the impact of noise in the image points. Tables 4.11 to 4.13 show the reconstruction and calibration errors due to a Gaussian noise added to the image points. The results show that matching error is relatively important in finding RMS reconstruction error.

Table 4.11: Effect of a Gaussian noise with $\delta = 0.5$.

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	10	1	0	0	8.13	3.20	2.90	0.00	0.00	3.12
$Using\Omega^*_\infty$	800	10	1	256	256	2.13	0.38	0.67	0.03	0.01	0.75
$Using\mathbf{E}$	800	10	1	256	256	4.38	1.48	0.39	0.01	0.10	2.18

Table 4.12: Effect of a Gaussian noise with $\delta = 1.0$.

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	10	1	0	0	12.29	4.19	3.82	0.00	0.00	3.71
$Using\Omega^*_\infty$	800	10	1	256	256	2.35	0.39	0.02	1.90	1.02	0.91
$Using\mathbf{E}$	800	10	1	256	256	6.21	3.10	0.11	3.90	6.64	2.94

Table 4.13: Effect of a Gaussian noise with $\delta = 1.5$.

Method	True values					Error in (%)					RE_{rms}
	α_x	γ	λ	x_0	y_0	α_x	γ	λ	x_0	y_0	
Linear	800	10	1	0	0	34.48	12.10	1.78	9.54	1.35	4.86
$Using\Omega^*_\infty$	800	10	1	256	256	3.68	1.71	0.12	2.30	2.41	1.45
$Using\mathbf{E}$	800	10	1	256	256	11.99	2.85	0.90	2.91	5.75	3.21

4.4 Discussion

We present experimental results on the sensitivity of reconstruction error to the assumptions on intrinsic camera parameters and to the noise in the input images. The significance of this experience is an experimental verification of the method that we propose for degeneracy. To study the sensitivity, three auto-calibration methods are used: auto-calibration using absolute quadric in linear and non-linear cases, and auto-calibration using essential matrix. We consider the effect of change in each camera parameter separately. The results show that a small deviation from zero in skew parameter has no major effect in the reconstruction and in estimating the remaining camera parameters. This fact is important, for several reasons. First, in most of the contemporary physical cameras, the skew is either zero or very close to zero; therefore assuming this parameter to be zero is safe and even in rare cases where it is not zero, the reconstruction error is small. Second, Triggs showed that a vanishing skew value could help in solving some degenerate cases due to critical motions [32]. Finally, given sufficient number of images, auto-calibration may be performed using only known skew parameter [33]. On the other hand, invalid assumptions in the values of aspect ratio and position of principal point generally cause major errors in the reconstruction. This is clear from the results obtained by linear method, which is based on the assumption that principal point is at the origin. Our conclusions conform to the theoretical discussion given by Hartley and Kaucic [29] where they report that the assumed position of the principal point may have a large effect on the reconstruction. From the application point of view, incorporating additional assumptions in all of the methods considered in this study is possible without difficulty. Linear method needs no initial values; however it is more sensitive to the input data. We have even experienced cases in which this method generated no feasible solution. Each method can be implemented by using various algorithms. There exist several solution methods to auto-calibration using essential matrix and we have adopted the method proposed by Cipolla [35] . Similarly

for the method based on absolute quadric, Triggs in [17] reports the results from two different methods used for solving the non-linear equation. Furthermore, Hartley mentions a method based on minimizing geometric error for calibrating a camera using absolute quadric [2]. However, we can state that in general the rate of sensitivity in the calibration parameters remains the same whatever the implementation method is. The output from these methods can be used as an initial point in iterative geometric error minimization methods such as bundle adjustment [16]. The relationship between the assumptions on camera parameters and reliability of the computed initial points for bundle adjustment can be verified in a similar method. In addition the evaluation method discussed here can be extended to include the effects of other physical properties of the camera such as distortions generated by camera lens.

CHAPTER 5

Face Model Generation and Refinement Approaches

5.1 Introduction

Appropriate combinations of geometric descriptors, surface texture and animation capabilities permit modeling of human faces. This chapter investigates modeling of human faces. Across the human population, the faces of individuals exhibit a lot of variation in their appearance, but they all still have a good deal of structure in common. Human faces are the most important feature distinguishing people from each other. They also play an important role in a wide range of applications. Graphical modeling of faces has obvious applications in the entertainment industry for character animation and in simulations involving people. Man-machine interaction benefits from the face tracking and the ability to monitor a user's attention and reactions automatically. Other applications range from interactive entertainment to security. In this chapter, a literature survey about the techniques for 3D modeling such as mesh deformation and Voronoi-Delaunay diagrams are presented. We also discuss some of the available methods for geometric description of faces. Each modeling method

is considered from two aspects: ability to support facial motions and ease of visualization.

5.2 Model-based techniques

Using high quality and high fidelity models is a challenging task for computer vision. In the absence of knowledge about the objects being observed, vision techniques often extract information for each image pixel. Improvements to this individual pixel view exploit the spatial coherency expected in most situations. Of course, such assumptions carry with them the added complexity of segmentation and finding the boundaries of coherent regions in space. Meanwhile, in the presence of knowledge concerning the observed objects, such as membership in a particular class of objects, model-based vision techniques extract information for each degree of freedom of a particular model. However, the use of model-based techniques introduces a new set of problems. The most significant one is the problem of maintaining alignment of the model with the image. Even though an accurate observation model might be available, if its current state does not correspond with the current state of the observed object, the advantage of using a model is lost. Another difficulty arises from dependencies between parameters. This problem becomes more important as model complexity increases. As a result, information extracted in a model-based framework must be used carefully, and combined in a way that considers these dependencies. The model-based technique described in this thesis is applied to faces. Face identification in 3D is a particularly natural test bed for our research.

5.3 Face modeling

Various methods have been employed in applying model based techniques to human faces. Each method however, should consider geometry, appearance and motion capabilities of the resulting model. In this section we discuss each part

separately and introduce some of the methods used in them. 3D face modeling involves three basic parts:

1. geometrical modeling
2. surface modeling
3. deformation modeling

Geometrical modeling defines the outer boundaries of the space occupied by the head and face being modeled. Three different geometric modeling methods have been used for facial applications. These methods involve:

1. using a group of discrete 3D points sampled from the outer surface of the head and face,
2. using a set of primitive geometric shapes like cylinders and spheres,
3. using a parametric mathematical model such as NURBS or B-Splines that defines the surface of the head and face.

In the first group of methods, 3D sample points are grouped into polygons (generally triangles) and the head is given by a mesh of these polygons. The main advantage of using a mesh is its simplicity of implementation and availability of hardware display facilities for effective rendering. These meshes also provide linear interpolated values for unavailable data points. This group of methods are discussed in more detail in Section 4.4

Primitive geometric shapes generally referred to as constructive solid geometry, (CSG), are too simple to produce a complicated shape like human face reasonably so in most cases the applications of this method are restricted to simple games or animations. Among the applications of CGS in shape modeling, we may mention medical volume visualizations. Voxels from medical images such as MRI are used for shape representation but due to their high resolution, rendering and animation are computationally intensive. Animation of the models created

by this way is complex when the motions include structural changes. However, some techniques such as marching cubes algorithm can be used to extract the outer surface of the model defined by the voxels. Parametric models have the advantage of creating a smooth surface which is determined using only a limited number of control points. However, in locations with high density details such as eyes and mouth, they suffer from a loss of accuracy. The parametric models can be grouped as implicit surfaces and parametric surfaces. An implicit surface is defined by a function such as $F(x, y, z)$ which assigns a scalar value to each point in 3D space. The surface defined by such a function is the set of points satisfying $F(x, y, z) = 0$. An example can be a sphere centered at $(0, 1, -1)$ with unit radius and defined using the following function.

$$F(x, y, z) = x^2 + (y - 1)^2 + (z + 1)^2 - 1$$

Polynomial functions may be used to describe implicit surfaces. The implicit surfaces defined this way can be also combined by blending them. Adding constraints to the functions used for implicit surface definition lets us define complex surfaces however; low degree of user control over them makes their usage in facial animation difficult. Parametric surfaces are generated by three functions of two parametric variables, one function for each of the spatial dimensions. These functions are generally based on quadratic or cubic polynomials and are usually defined in terms of control values and basis functions. Most famous functions in this group are *B-splines*, *Bezier* and *NURBS*. A general expression for these functions can be

$$S(u, v) = \sum_i \sum_j V_{ij} B_{ik}(u) B_{jm}(v)$$

where $S(u, v)$ is the parametric surface, V_{ij} are the control values, and $B_{ik}(u)$ and $B_{jm}(v)$ are the basis functions of polynomials of orders k and m respectively. To add these details hierarchical B-Splines have been used. Surface modeling refers to adding color and texture values to each point on the surface defined by geometric modeling.

Deformation modeling in used to specify the type of surface and geometry changes due to expressions such as anger and laugh or activities like talking and eating. These changes are the basic part of animation systems. Form, size and location of head bones and muscles and the joints connecting them are among the features which should be considered in deformation modeling.

5.4 Geometric modeling using 3D sample points

Geometric models of head and faces may be acquired from 3D sample points. [34, 40, 47] These points may be found from direct measurements on the head and face such as by range scan data, or obtained from the images taken by digital cameras. In both cases, the number of the sample points is not sufficient for creating a smooth detailed surface and an interpolating process is necessary. To accelerate the rendering process, sample points are generally converted into a triangular mesh, which at the same time plays the role of interpolating step. One of the major problems here is that in range scanned data, large number of points makes rendering process and animation unacceptably slow. On the other hand, number of sample points obtained from 2D image is generally low. This is partially due to lack of texture and partially due to low quality of images or illumination problems. In addition, defining the boundaries of face features especially in concave regions can make artifacts. A generic mesh with a limited number of vertices can solve the problems in both cases. Assuming a generic mesh and deforming it using the data obtained from range scanners or 2D images implies that all face and head models are geometrically similar to each other and the differences are very slight. However, the correlation between the face features and the relationships in the geometric properties of the vertex points makes the morphing of a generic mesh into a specific one a difficult task. This is the main argument in favor of the methods, which prefer creation a new mesh to deforming a generic one. Considering the above discussions, we may group face modeling algorithms using 3D sample points as:

1. generic mesh deforming algorithms,

2. interpolating algorithms,

3. mesh creating algorithms.

In the first group, the locations of vertices in the generic mesh are changed by moving them towards the sampled points. Remaining vertices are adjusted using either an interpolation algorithm or simulating deformations as external forces. Akimoto *et al* have used a generic mesh, which includes two types of vertices: feature vertices and non-feature vertices [42]. Feature vertices are moved to their new positions obtained from a frontal and a profile images. Non-feature vertices are displaced as much as a vector unit computed by interpolating the displacement vectors of feature vertices. A coefficient value function is used in the interpolation, which gives large values for small distances between vertices. This function emphasizes on the importance of nearby feature vertices. Luo and King deform a generic mesh in two stages [43]. The first stage includes fixing the location of boundary vertices and displacing inner vertices by interpolating their location values. In the second stage, the vertices from the boundaries of face features are found and adjusted. Yan *et al* have considered a mesh with springs connecting the vertices to each other [44]. Deformation in their algorithm is done in two levels. In the first level which they call it as coarse level, a set of vertices are allowed to move. This may consist of an organ or part of an organ. The movement then is propagated in the relative area. In the fine level, a single vertex of the mesh is moved to a new place. The 3D coordinates of the neighboring vertices are computed by deformation algorithm as in the coarse level. Tang and Huang used a generic mesh and two images of a person in frontal and profile positions to create the 3D model of a person. They deform the mesh by first globally mapping it to fit the face boundaries. Then each facial feature is mapped locally to fit the feature points. They finally combined face models mapped to different views of the face images and got a complete 3D face

model.

Second group argues that a specific head is a weighted average of a group of base head models. The only unknown part here is the estimation of weight coefficients. Generally these methods are restricted to a limited form of faces like a specific race. Among this group Zhang [20] and Vetter [45] have carried out the most popular studies. Zhang defines a face as a linear combination of a neutral face and some number of face metrics where a metric is a vector that linearly deforms a face in certain way, such as to make the head wider or the nose bigger. A triangular mesh gives the neutral face. In fitting the face model, a search is done for both the pose of the face and the metric coefficients to minimize the distance from the reconstructed 3D points to the face mesh.

Third group is based on the argument that a mesh can always be created whenever a set of 3D vertices is available. Generally these groups of algorithms use generic surface modeling methods such as Delaunay triangulation. The main disadvantage of these methods when applied to human face modeling with passive data acquisition is the uncertainty in locating main feature vertices. This is generally compensated for by starting from a generic mesh and using the 3D coordinates of the vertices from the generic mesh whenever no data is available about a feature point. In this group we may mention the work by Thalman *et al* [46], Fua *et al* [9]. Thalman *et al* start with mapping the vertices of a generic mesh on a 2D plane. Some of these points corresponding to main features of the face are picked which are called control points. A Delaunay triangulation is created and non-control points are expressed using their local barycentric coordinates. Next step in their algorithm is moving the control points to their correct locations found in previous steps. When the control points are placed, the displacements of non-control points are interpolated according to their local barycentric coordinates. This algorithm assumes accurate detection of the location of control points. It also suffers from the problem of losing accuracy in highly detailed areas. Fua *et al* start with an initial mesh, which is given by tri-

angulating the depth map provided in the previous step. Then they iteratively refine the initial mesh by minimizing an energy function consisting of an internal term and an external term. The external term is derived from the stereo information assuming that the projections of a 3D point on 2D image planes have the same intensities. This minimizes the intensity difference between the projections of the face. The internal term tends to minimize the deviation of the mesh from its regular form by minimizing the global curvature of the surface. However since the parameter space of the energy function is too large, they simplify the mesh by finding the direction of minimum and maximum curvatures and increasing the length of the edges of the new mesh in the direction of minimum curvature. This algorithm is based on the assumption that initial locations of the points are accurate enough to provide the reliable values for curvatures in the initial mesh.

5.5 Voronoi-Delaunay diagrams

In all of the methods dealing with meshes, artifacts due to improper location of vertices, overlapping or encroaching of edges, existence of skinny polygons, etc can deteriorate the models. A well known and mathematically sound method for modeling is given by Voronoi-Delaunay diagrams [48]. Voronoi-Delaunay diagrams have significant importance in generating 3D models. In this section, the basic underlying theory is introduced. The section includes the necessary definitions, theorems and algorithms for creating, modifying and updating Delaunay meshes. The theoretical concepts provided here are used in describing our mesh deformation algorithm in Chapter 6.

Let $\mathbf{P} = \{p_1, p_2, \cdots, p_n\}$ be a set of points in Euclidean 2D space. These points are called sites and the plane is partitioned so that each point is assigned to its nearest site. All points assigned to each partition such as p form the Voronoi region or Voronoi polygon $\bigvee(p)$. The set of all points that have more than one nearest neighbor is called Voronoi diagram for the set of sites. Given

a set $\mathbf{P} = \{p_1, p_2, \cdots, p_n\}$ of n points in the plane and the Voronoi diagram $\bigvee(p)$, an undirected graph can be created by using points of $\mathbf{P}$ as vertices of the graph and connecting vertex p_i to vertex p_j iff the regions V_i and V_j share an edge. The resulting graph is a triangulation of the convex hull of $\mathbf{P}$ having $\mathbf{P}$ as its set of vertices. This definition can be extended to any set of points in $\mathbb{R}^m$ where the convex hull of the set is associated with its set of vertices. Let $\mathbb{E}$ be any normal affine space. Given any *n+1* affinely points a_0,$\cdots$,a_n in $\mathbb{E}$, the n-simplex or simplex defined by a_0,$\cdots$,a_n is the convex hull of these points. n is called the dimension of the simplex and a_0,$\cdots$,a_n are its vertices. A simplicial complex or in short a complex in $\mathbb{E}^m$ is a set $\mathbf{K}$ consisting of a set of simplices in $\mathbb{E}^m$ satisfying the following conditions:

- every face of a simplex in $\mathbf{K}$ also belongs to $\mathbf{K}$
- for any two simplices σ_1 and σ_2 in $\mathbf{K}$ if $\sigma_1 \bigcap \sigma_2 \neq \emptyset$ then $\sigma_1 \bigcap \sigma_2$ is a common face of both σ_1 and σ_1

Second condition guarantees that the various simplices forming a complex are connected nicely.

A polytope is the geometric realization of some simplicial complex. A polytope of dimension one is called a polygon and a polytope of dimension two is called a polyhedron. It can be shown that each region of a Voronoi diagram is a convex polytope. Figure 5.1 shows a Delaunay triangulation. It is clear from Figure 5.1 that Delaunay triangulation of a set of points in $\mathbb{E}^m$ is also the convex hull of them. This property and two other important properties of Delaunay triangulations can be formally stated as follows.

The boundary of the Delaunay triangulation of a set of points such as $\mathbf{P}$ is also their convex hull.

A triangulation $\mathbf{T}$ associated with $\mathbf{P}$ is the Delaunay triangulation of $\mathbf{P}$ iff every *(m-1)*-open ball circumscribed about an m-simplex contains no other point from $\mathbf{P}$. A Delaunay triangulation maximizes the minimum angle of the trian-

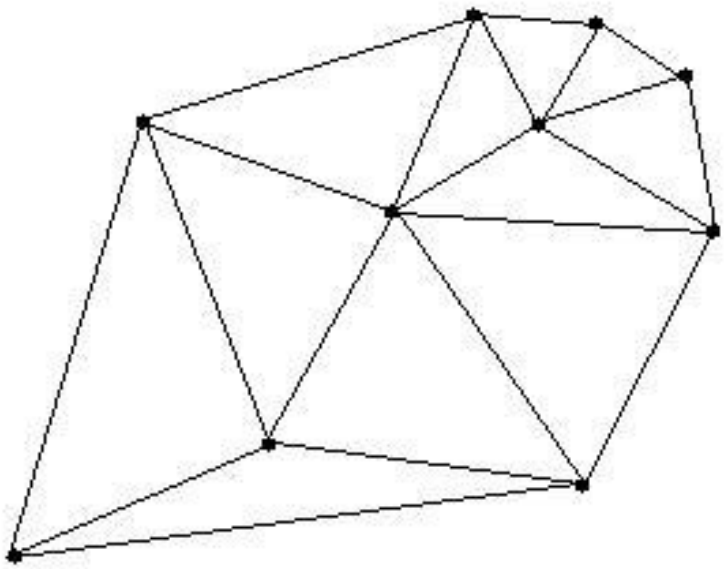

Figure 5.1: A sample Delaunay triangulation.

gles involved in any triangulation of a set of points **P**. Voronoi and Delaunay diagrams have many applications such as motion planning, finding minimum spanning trees, finding largest empty cycles and mesh generations.

5.5.1 Mesh Refinement

Adaptive mesh refinement places more grid points in areas where the error in the solution is known or suspected to be large. Local error estimates based on a solution computed on an initial mesh are known and can be used to determine which elements should be refined. One approach to mesh refinement iteratively inserts extra vertices into the triangulation, typically at edge bisectors or triangle circumcenters which is the center of a triangle's circumcircle. Insertion may be followed by mesh smoothing. This approach gives a finer mesh, but not an edge conforming refinement of the original mesh. A mesh refinement is called edge conforming if it includes the boundaries of the original triangles. To compute such a refinement, another approach is used. This approach splits triangles in

need of refinement, by adding the midpoints of sides. The algorithm below gives the overall approach.

Algorithm 2 Triangle splitting algorithm.

---

1 $k = 0$

2 estimate the error on each triangle in initial mesh T_0

3 while maximum error on a triangle is larger than the given tolerance **do**

3.a based on error estimates, mark a set of triangles S_k to refine

3.b divide the triangles in S_k, and any other triangles necessary to form T_{k+1}

3.c re-estimate error on each triangle of T_{k+1}

3.d $k = k + 1$

4 end while

There are a number of alternatives for step 2, in which the current mesh T_k is adaptively refined. In regular refinement, the midpoints of the sides of a marked triangle are connected, as in Figure 5.3, to form four similar triangles. Unmarked triangles that received two or three midpoints are split in the same way. Unmarked triangles that received only one midpoint are bisected by connecting the midpoint to the opposite vertex as in Figure 5.3. Before the next iteration of the algorithm, bisected triangles are connected together and marked for refinement; this precaution guarantees that each triangle in T_{k+1} will either be similar to a triangle in T_0 or be the bisection of a triangle similar to a triangle in T_0. Hence regular refinement, regardless of the number of times through the refinement loop, produces a mesh with minimum angle at least half the minimum angle in T_0.

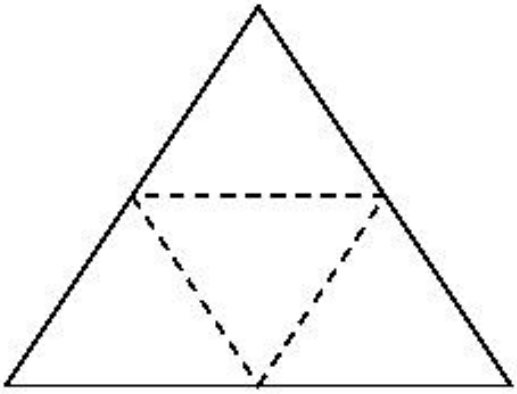

Figure 5.2: Splitting a triangle by four.

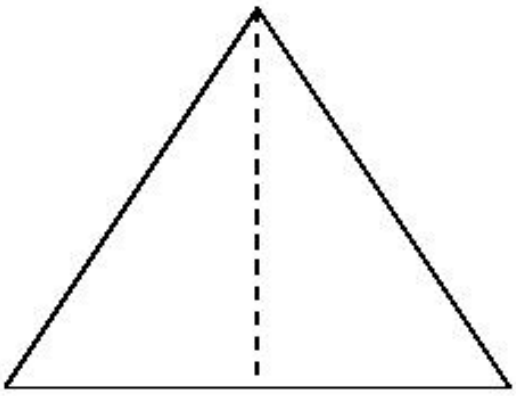

Figure 5.3: Splitting a triangle by two.

Figure 5.4: Smoothing algorithm applied to a mesh.

5.5.2 Mesh Smoothing

Mesh smoothing adjusts the locations of mesh vertices in order to improve element shapes and overall mesh quality. Mesh smoothing has a certain advantage over other mesh improvement methods. In mesh smoothing the topology of the mesh remains invariant, therefore it preserves important data structures. Laplacian smoothing is the most commonly used smoothing technique. This method sweeps over the entire mesh several times, repeatedly moving each adjustable vertex to the arithmetic average of the vertices adjacent to it. A variation weighs each adjacent vertex by the total area of the elements around it. Laplacian smoothing is computationally inexpensive, but it does not in general guarantee improvement in element quality. In fact, Laplacian smoothing can even invert an element, unless the algorithm performs an explicit check before moving a vertex.

Another class of smoothing algorithms uses optimization techniques to determine new vertex locations. Both global and local optimization-based smoothing offer guaranteed mesh improvement and validity. Global techniques simultaneously adjust all unconstrained vertices; such an approach involves an optimization problem as large as the number of unconstrained vertices is computationally expensive. Local techniques adjusts vertices one by one or an independent set of vertices in parallel resulting in a cost more comparable to Laplacian smoothing. Figure 5.4 shows the results of a local optimization-based smoothing algorithm applied to a mesh generated adaptively. The mesh on the left is generated using the bisection algorithm for refinement; the edges from the coarse mesh are still evident after many levels of refinement. The mesh on the right is generated by a similar algorithm, only with vertex locations optimized after each refinement step.

CHAPTER 6

Generating Human Face models

Head modeling should be considered as a special case in the general problem of 3D object modeling. The reason is two fold. On one hand the symmetric form of human face and the subtle anthropological relations between facial features makes even slight deformations unacceptable. On the other hand, the geometric properties of human faces are very near to each other. This means that creating the face model of a specific person needs very accurate measurements. One solution to this problem which makes use of the similarities between different faces and their anthropological properties is starting with a generic model. This model may be considered as the average geometric model of distinct faces. Our algorithm takes advantage of this fact and tries to deform a generic model to a specific face. The deformation process requires local re-triangulation in the areas with high curvatures. This is achieved by locally applying Delaunay triangulation method.

6.1 Geometric Head Modeling

This section describes in detail the proposed algorithm for geometric modeling of human faces [11]. Our algorithm has the following features.

Input:

- Two sets S_1 and S_2 of 3D points from the surface of the face. First group of points are labelled with the identification number of the related facial feature and the position of the point in the feature. As an example a point in this group may be given as $\mathbf{p}(Nose, Tip)$. Second group of points are not labeled.
- A generic model of the human face in the form of a 3D mesh.
- Texture values given in cylindrical format.

Output: Textured 3D model of a specific face.
The input points are obtained in two stages. In the first stage 9 points of the facial features are located in the images and their 3D positions are obtained. These points are shown in Figure 6.1. The points in set S_1 are:

1. $\mathbf{p}(Forehead, Top)$
2. $\mathbf{p}(LeftEye, LeftCorner)$
3. $\mathbf{p}(LeftEye, RightCorner)$
4. $\mathbf{p}(RightEye, LeftCorner)$
5. $\mathbf{p}(RightEye, RightCorner)$
6. $\mathbf{p}(Nose, Tip)$
7. $\mathbf{p}(Lips, LeftCorner)$
8. $\mathbf{p}(Nose, RightConer)$
9. $\mathbf{p}(Chin, Tip)$

Figure 6.1: Location of feature points.

This step involves locating and matching the feature points and their reconstruction. Locating the feature points is done manually in two images and then these locations are refined by searching in a 7×7 window around each point and finding best matches using normalized cross-correlation. The normalized cross correlation of the points in each iteration is found from Equation 6.1.

$$\frac{\sum_{x,y}[f(x,y) - \overline{f}_{u,v}][g(x-u,y-v) - \overline{g}_{u,v}]}{\{\sum_{x,y}[f(x,y) - \overline{f}_{u,v}]^2 \sum_{x,y}[g(x-u,y-v) - \overline{g}_{u,v}]^2\}^{0.5}} \tag{6.1}$$

Second set of input points S_2 are random points from facial areas in the images. The points in this group are obtained by applying an interest point detector (Harris corner detector) and then finding the corresponding point in the second image and reconstructing the 3D point. Harris corner detector [49] is given in Equation 6.2.

$$R = Det(\mathbf{M}) - kTrace(\mathbf{M})^2 \tag{6.2}$$

Here R is the corner response value which shows the possibility of existence of a corner point at each pixel, K is a constant which is fixed by Harris to $k = 0.05$ and $\mathbf{M}$ is a 2×2 matrix defined as

$$\begin{bmatrix} \frac{\partial I}{\partial x} \otimes W & \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \otimes W \\ \frac{\partial I}{\partial x}\frac{\partial I}{\partial y} \otimes W & \frac{\partial I}{\partial y} \otimes W \end{bmatrix}$$

where W is a sliding window, I is the image and $\otimes$ denotes convolution operation. The geometric deformation of the generic mesh is done in five steps. In the first two steps the generic mesh is scaled so that each face segment between facial features corresponds to the size of the same region in the images. Step 3 and 4 deforms features and face surface and step five refines the mesh. The five steps in details are as follows.

1. Global scaling the generic mesh.

2. Local scaling the generic mesh.

3. Coarse deformation.

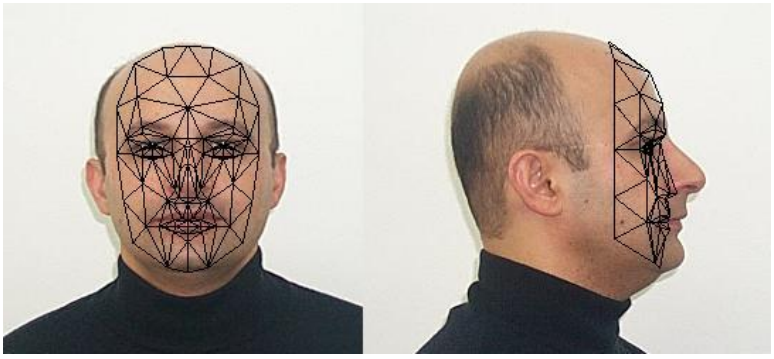

Figure 6.2: Global scaling (Step one).

4. Fine deformation.

5. Mesh smoothing.

Since the mesh model and the input points may be in different sizes, the first step performs a scaling operation based on the maximum and minimum coordinate values of the points in the input set. No local displacement of the facial features is done here. Figure 7.4 illustrates this step.

Facial feature sizes and locations are different in each face case. The generic mesh is divided into four layers which are separated by the corners of lips, nose tip, and corners of eyes. The vertices corresponding to these boundary points are displaced to match their true values. The locations of the remaining intermediate vertices are computed by interpolating over the displacement of the boundary vertices. The values of the boundary vertices are given as the first input set of 3D points. Local scaling carried out in this step is shown in Figure 7.5. In this step the 3D points from the first group are used to deform the generic mesh. In this step since the points are labeled, we displace the corresponding feature vertices and bring them to the locations given by these input points. Two main issues are considered in this displacement:

1. Anthropological relationship between face features should be preserved.

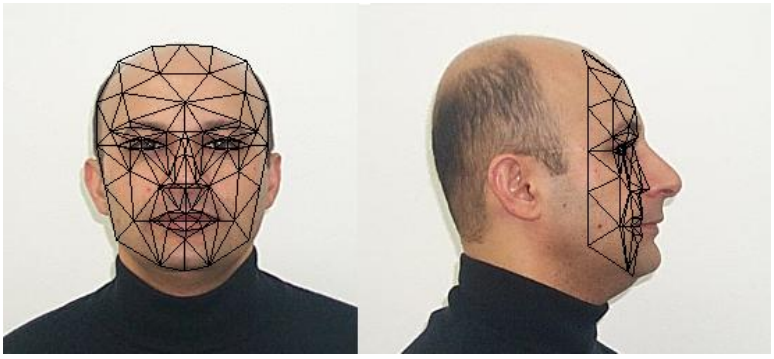

Figure 6.3: Local scaling (Step two).

2. Displacing a vertex should not create unrealistic warps such as corners with sharp curvatures.

To impose both restrictions we are considering the mesh as an interconnected set of vertices in which displacing a vertex affects other (specially neighboring) vertices. Therefore moving a facial feature or one of its vertices in a direction imposes a force on all vertices connected to it drawing them in the same direction. This force is simulated by considering the links connecting the vertices as massless spring in equilibrium states. Total force generated by the springs, ξ around a vertex determines its new position. Our aim by introducing the spring term is avoiding sharp sparks in locations with high rate of change and small edges. However, unwanted warps in the mesh can not be avoided by just considering the spring like connections between vertices. In this specific mesh deformation application we have the advantage of knowing the general form of the face *a-priori*. Therefore to preserve the topology of the mesh we have added a second restriction using the principal curvatures at a given point on a surface which measures the maximum and minimum bending of the surface at that point. To find out the principal curvatures, we compare the normals to facets about a vertex and the minimum and maximum angles between these normals define the curvature (ζ).

$$\zeta = Max(Diff(n_i, n_j))$$

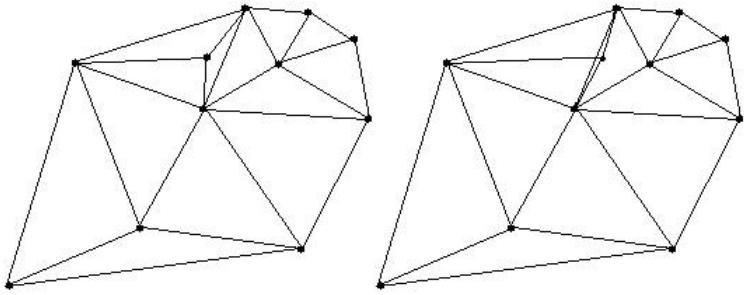

Figure 6.4: Degeneracy due to encroaching an edge by another one.

Where n_k is a normal about vertex n. Considering the both issues mentioned above, a cost function is defined to adjust the location of other vertices in the mesh whenever a vertex is moved to a new position. The cost C function is defined as:

$$C = \Delta(\zeta) + \Delta(\xi)$$

Drawing a vertex to a new position may cause encroaching on a link by another one. This makes small concave holes and artificial crease lines when mapping texture to facets. Figure 6.4 shows this type of degeneracy.

Our solution to this degeneracy is the a gradual displacement of each node and propagating the displacement to the neighboring nodes. For each point in the second set of the 3D points, a similar deformation is necessary. However in this step we are not sure about the vertex correspondences between new points and mesh vertices. Furthermore there may be no corresponding vertex for a 3D point at all. The latter case requires a re-triangulation in that part of the mesh. Therefore this step starts with a search for finding the nearest vertex to the position of a 3D point from the second set. In this step, we do not expect a drastic change in the form of the generic mesh because it has been already undertaken three deformation steps and the new points should add slight modifications. We are considering a threshold in the distance between the 3D point and the nearest vertex. This threshold value bans large changes in the mesh. If the 3D point is within the open ball $\mathbf{B}(k, t)$ defined by the nearest vertex $\mathbf{k}$ and threshold

t, then the vertex is moved towards the 3D point using the same algorithm as used in step 3. However, the point may be near to the outer surface of the mesh but far from any vertex in it. This generally happens when a surface with low curvature has been linearly approximated by a large facet which gets far from the real surface in mid parts of the facet. Hence the new 3D points aim at refining the surface by a better approximation of the volume space occupied by the mesh. In these cases we are inserting the new point as a new vertex to the mesh. The insertion however, considers several possible cases depending on the location of the vertex and topology of the mesh in that region. The algorithm to add a vertex to the mesh includes:

Algorithm 3 Finding Delaunay cavity.

---

1 For each 3D point t **do**

1.a Find all facets k which includes the mapping of t in their diametral ball $\mathbf{D}(k)$ defined as the open ball of minimal diameter containing the vertices of k.

1.b Find the Delaunay cavity form the union of the facets obtained in step **1.a** when the common edges are removed.

2 End For

Figure 6.5) shows the Delaunay cavity. Re-triangulation of the mesh is done by adding edges ¡t,m¿ where m is a vertex from the Delaunay cavity. This triangulation preserves the slight changing rate in the curvature value when the new vertex is an almost plain area as mentioned above. In the case of the regions with high frequency, a threshold value is used to reject points far from the surface of the mesh. This threshold value is defined using radius-edge ratio of the facets falling in the Dealunay cavity of that point. Radius-edge ratio of a facet k is defined as the ratio of the length of the radius of $\mathbf{D}(k)$ to

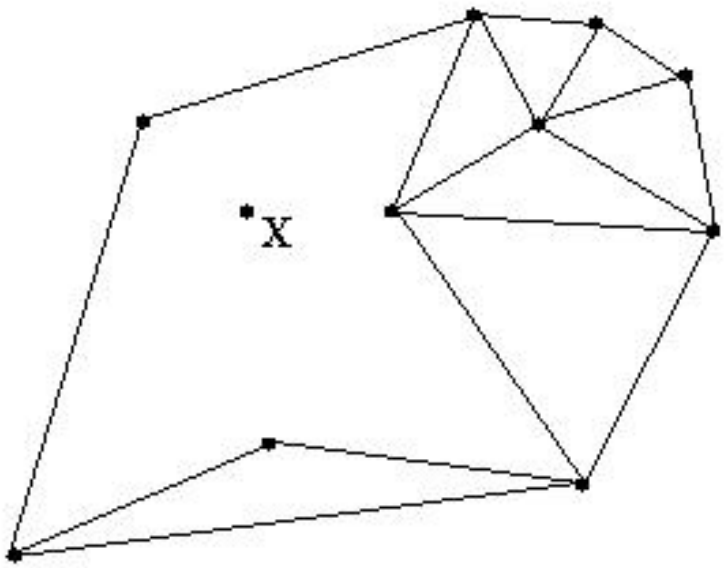

Figure 6.5: Delaunay cavity.

the length of the shortest edge of k. Two main degenerate cases may happen in this type of triangulations. The first degenerate case has been shown in Figure 6.6. This case occurs when an edge added during re-triangulation of a Delaunay cavity encroaches on a boundary edge of the cavity. It happens when the vertex is not visible from the new vertex. This can be stated as follows. If an empty ball **B** contains n points in its boundary then the n-simplex formed from their convex hull is present in their Delaunay triangulation. However in degenerate cases the empty ball criterion does not determine a triangulation of the input points. Instead it defines decomposition as a unique collection of Delaunay protopes. Second case happens when the sight line from t to a cavity boundary vertex pierces the facets of the cavity area. This case has been shown in Figure 6.7. For the first group of degenerate cases we have changed the definition of Delaunay cavity to include only those points in its boundary that does not cause an edge encroaching while triangulation. This also preserves the relationship between convex hull and the Delaunay cavity. The modified cavity and re-triangulation are shown in Figure 6.8. Second

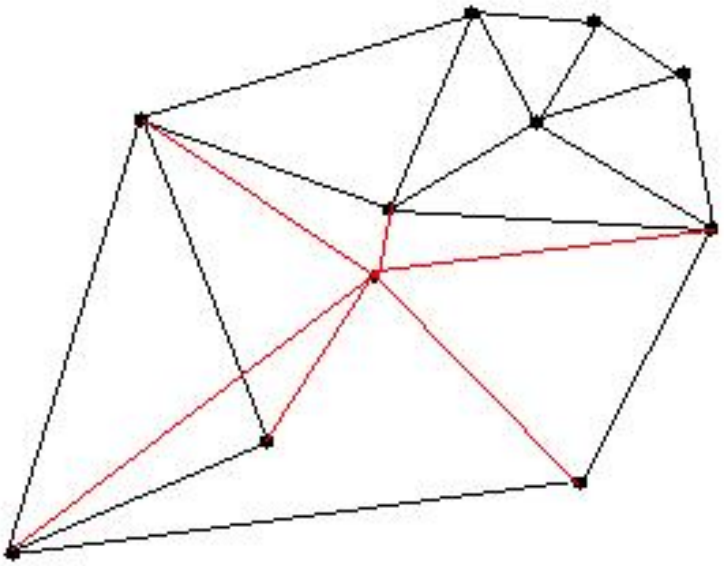

Figure 6.6: Degeneracy in re-triangulating Delaunay cavity .

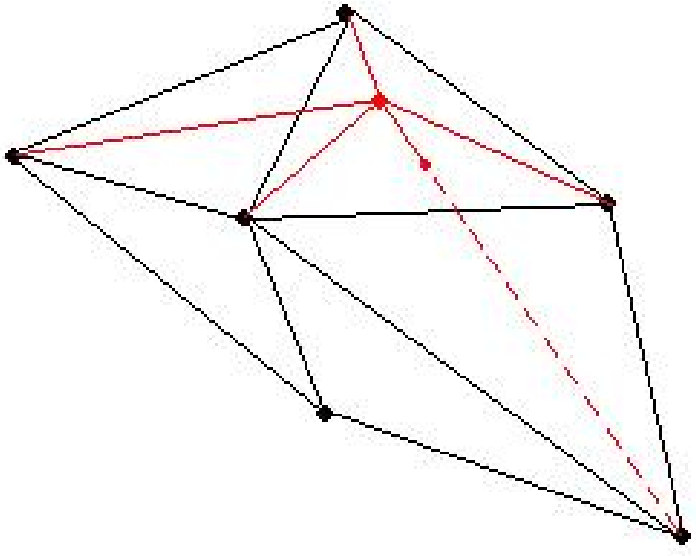

Figure 6.7: Degeneracy due to piercing a polygon by an edge.

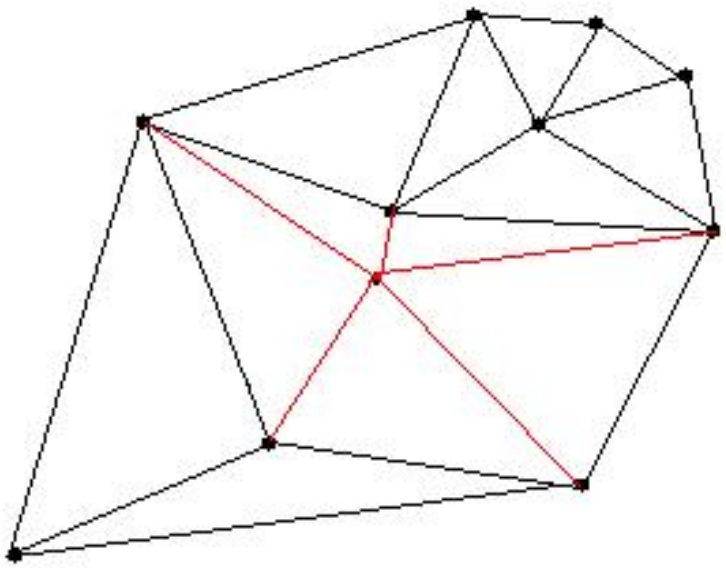

Figure 6.8: Modified Delaunay cavity to remove degeneracy.

case of degeneracy affects the topology of the mesh. Re-triangulation in this case violates the boundaries of the local deformations. Most unwanted effect of these deformations is creating concave areas in a convex sub-region. Our definition of the 3D Delaunay cavity also prohibits this type of degeneracy. An edge in Delaunay triangulation can not be encroached by edges of the mesh. Figure 6.9 shows an example of this case and our re-definition of the Delaunay cavity. Re-triangulation and displacing vertices may create some small and skinny facets. Skinny and small facets affects deformation algorithm and slows down rendering process. This step tries to eliminate these facets. To detect skinny facets the radius-edge ratio is used. This criteria is defined as the ratio of the length of the radius of diametral ball of each triangle to its shortest edge. Last processing done is a Laplacian smoothing applied to the deformed mesh to improve its quality. We have modified the Laplacian smoothing algorithm to also impose restrictions which define a Delaunay triangulation. This restriction can be impose by adjusting the location of each vertex through the barycentric its coordinates.

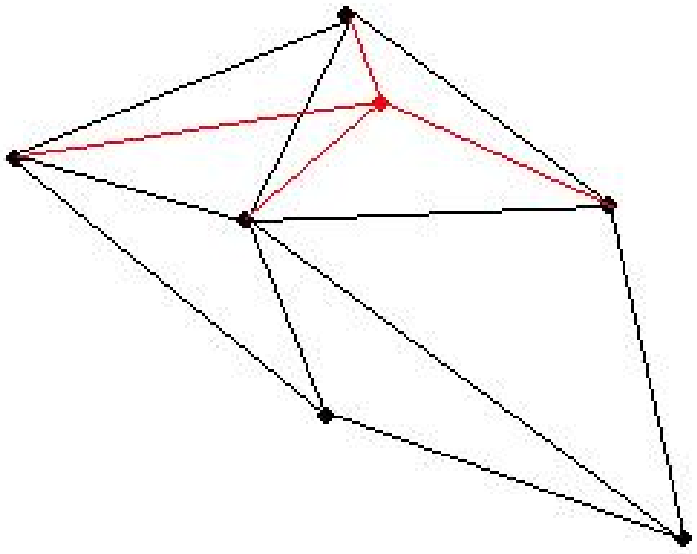

Figure 6.9: Removing second case of degeneracy.

6.2 Texture Mapping

Texture mapping is the last part of the human face modeling. Impressive visualization and realistic appearance of the model are dependent on this step. On the other hand a mesh is a linear estimation of a nonlinear volume and therefore non-realistic artifacts may be visible in the edges connecting faces to each other. We use an approach similar to Pighin et al.s method [50] to generate a view independent texture map. We also construct the texture map on a virtual cylinder enclosing the face model. For each vertex on the face mesh, we compute the blending weight of each image based on the angle between surface normal and the camera direction. If the vertex is invisible, its weight is set to 0.0. The weights are then normalized so that the sum of the weights over all the images is equal to 1.0. We then set the colors of the vertexes to be their weights, and use the rendered image of the cylindrical mapped mesh as the weight map.

CHAPTER 7

Experimental Results

In this chapter the experimental results of applying human face modeling algorithm which is introduced in Chapter 6 are given. The main steps of the algorithm are followed using a sample sequence of images and the deformations undertaken by the generic mesh are exhibited. Three more 3D models of human faces are displayed as the results of applying the algorithm to input images.

7.1 Mesh Deformation

The geometric model of a person in generated by gradual deformation of a generic mesh. The deformation is carried out using a cloud of 3D points obtained from a sequence of images. Figure 7.1 shows a sample sequence of input images. The 3D points used in deformation step are grouped as labeled points set S_1

Figure 7.1: A sample sequence of input images.

and random points set S_2. S_1 includes points from basic facial features. These

points are provided manually, however, their exact locations are obtained using corner detection and cross correlation as explained in Chapter 6. Figure 7.2 illustrates the set of labeled 3D points. To obtain second set of 3D points, Harris

Figure 7.2: Labeled set of 3D points.

corner detector is applied to the sequence and matching point pairs are found considering the disparity consistency and maximum cross correlation values [49]. Figure 7.3 illustrates the result of applying corner detection and matching the points. We use the 3D points obtained from the matching and reconstruction

Figure 7.3: Corner points and matched points pairs.

steps in deforming a generic mesh. We apply the deformation to two different meshes. The first mesh is a coarse one with 168 vertices and 275 faces. The dense mesh includes 1355 vertices and 1739 faces. The generic mesh should be scaled before any deformation. The uppermost and lowermost points from set S_1 are used in scaling the generic mesh. The scaled generic mesh superimposed on the front and side views in the sequence is shown in Figure 7.4. It is

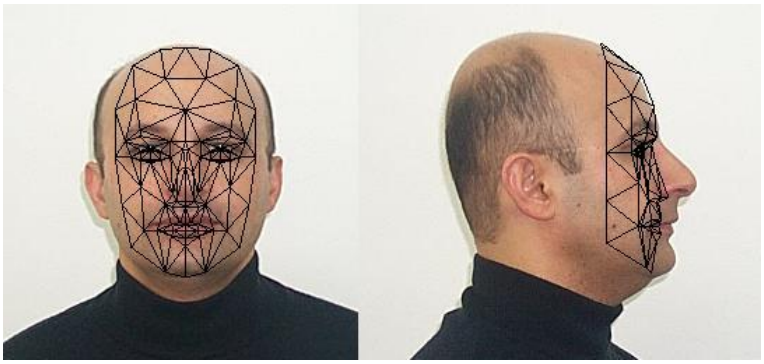

Figure 7.4: Scaling the generic mesh.

important to note that scaling does not bring the facial features to their proper locations. This stems from the fact that facial areas have different extent in different people. Drawing the facial features to their suitable locations and expanding and contracting facial areas such as forehead, chicks or chin are carried out in the next step. The 3D points from set $s1$ are used for local scaling facial areas and interpolating the location of vertices in each area. The result of local scaling of the generic mesh is shown in Figure 7.5. After local scaling the

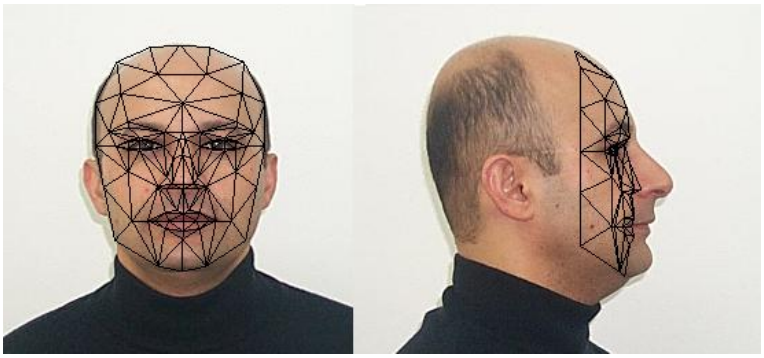

Figure 7.5: Local scaling and interpolating facial areas.

mesh, each facial feature is in its suitable location but still the size of facial features in in images and the generic mesh may differ. Pulling the main vertices of the facial features towards their locations in the input images and reshaping them is accomplished in the following step. The deformations are imposed to

the mesh by pulling or pushing neighboring vertices when a vertex is dislocated. Second set of 3D points are applied to the generic mesh causing displacement of available vertices and introduction of new ones. The deformed mesh is illustrated in Figure 7.6. In this experience we use the coarse mesh. The total number of

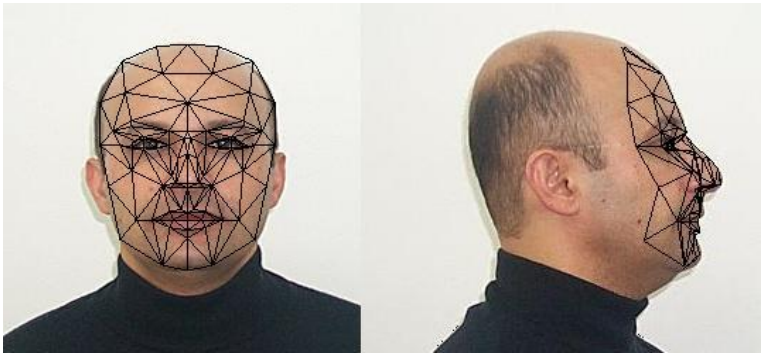

Figure 7.6: Refined mesh superimposed on front and side views.

points obtained form point pair matching step is different in each case. In this experience, 274 point pairs are detected but 31 points are rejected during the deformation due to their distance from the generic mesh. Only 29 points causes a face split in this experience. This number depends on the threshold value used in displacing a vertex and the density of vertices in the mesh. Last step in face modeling is covering the polygons with texture values. This step is the last step in 3D model generation. The textured model from different view points are shown in Figure 7.7. A second sequence of views from a model generated using our algorithm is illustrated in Figure 7.8.

7.2 Reliability of Deformation Algorithm

The generated 3D models presented in the mesh deformation section of experimental results, do not provide an objective reliability criteria for assessment of the algorithm. In this section we are considering the 3D data obtained from range scanning two different faces as ground truth and verifying the correctness

Figure 7.7: Textured model shown from different views.

Figure 7.8: Textured model views (Second example).

Figure 7.9: Textured model views (Third example).

of the results of our algorithm against them. The details of our method and numerical results are given in the following subsections.

7.2.1 Range Scan Data

The data obtained form range scanning two human faces are available in the form of a PLY file. A PLY file include both 3D values of vertices and polygons which relate vertices to each other. The files we are using here have 302948 vertices, 605086 polygons and 303380 vertices, 605902 polygons respectively. The range scan data of these faces have been rendered in Figures 7.10 and 7.11.

7.2.2 Test Setup

Our mesh deformation algorithm is based on two sets of 3D points. The first set $S1$ includes labeled set of facial feature points, and the second set $S2$ includes random points. In the testing process, these two sets of points are obtained by clicking on some points in the window of a developed rendering program and obtaining the coordinates of the vertices nearest to the mouse cursor tip. The generic mesh is deformed using these two sets of points. To verify the accuracy of deformation depth values of some points from different facial areas are compared in our model and range scanned data. To avoid any error in comparison we are connecting points form the first set $S1$ to each other and then dividing the connecting line by 4 and obtaining x and y coordinates of the testing points. The z coordinate values of these points are found from our model and range scan data and compared to each other. Figure 7.12 shows the location of testing points on the first sample image. The average error in each facial region is given in Table 7.1.

In these experiments we scale the y coordinate values of the vertices in the generic mesh to range between -1 and 1. x and z coordinates are scaled using the same factor. Considering this fact, 0.01 in error values means that the displacement

Table 7.1: Average error in face modeling using Cyberware data

	Eyes Region	Mouth Region	Cheeks Region
First Image	0.016	0.012	0.025
Second Image	0.014	0.013	0.017

is $\frac{1}{200}$ times of head height. The error values are higher in the first image. Since the generic mesh represents the face model of a male, it may be justified by the need for larger changes in the mesh compared to the second image. The reconstructed faces using the dense mesh are displayed in Figures 7.13 and 7.14 The experiments are also repeated for the total vertices. In this case, for each vertex from the cyberdata using its x and y coordinates, we map the point on the generic mesh and find the z value. The computed z values are compared with the ground truth value from the cyberdata. In this experiment we have the following considerations:

- Since some of the vertices from cyberdata which belong to hairs, nostrils, eyebrows and so on are not reliable, we have applied a threshold value to reject a point if the difference of its z with the z value coming from the mesh is too high. This threshold is obtained from the average difference of z values in the cheeks regions and is rounded up giving 0.03.

- The head model in generic mesh contains only the frontal part and therefore we are rejecting all points from cyberdata with a z value less than 0.25. This guarantees the elimination of the back side of the head.

- To prevent the cancellation of the values, we are considering the square of the differences. This has the benefit of closeness to a standard error metric namely RMS error.

The results of the second experiment are given in Table 7.2

Table 7.2: Average error in face modeling using Cyberware data

	Average Squared Error	RMS
First Image	3.15×10^{-4}	1.775×10^{-2}
Second Image	3.23×10^{-4}	1.797×10^{-2}

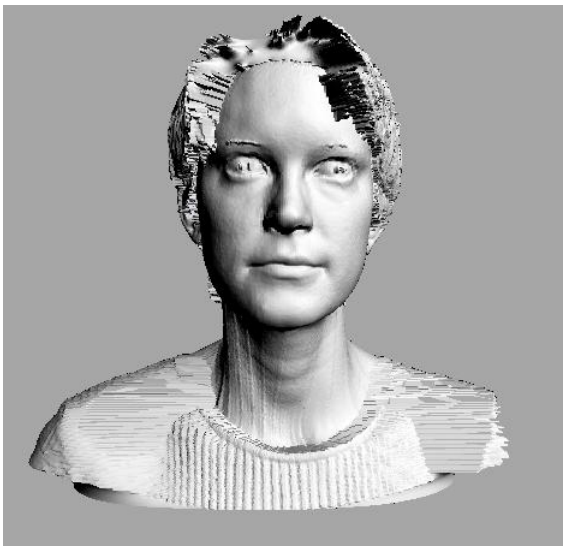

Figure 7.10: First example of range scan data.

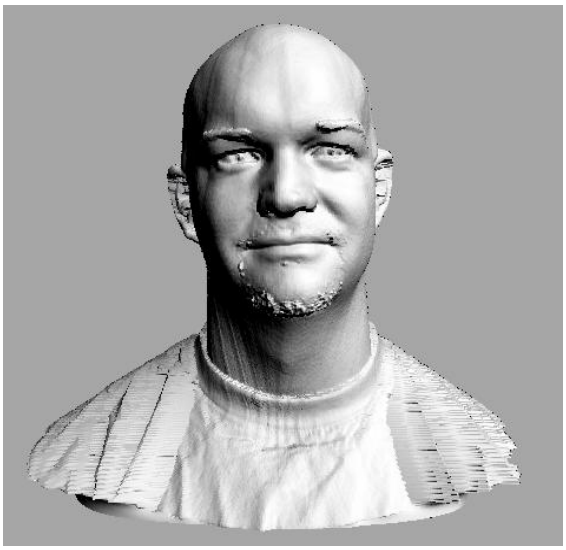

Figure 7.11: Second example of range scan data.

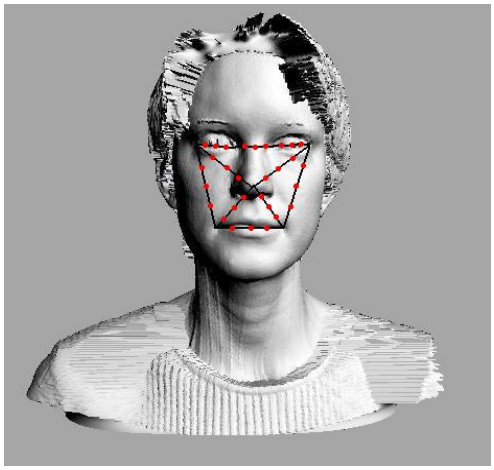

Figure 7.12: Location of testing points.

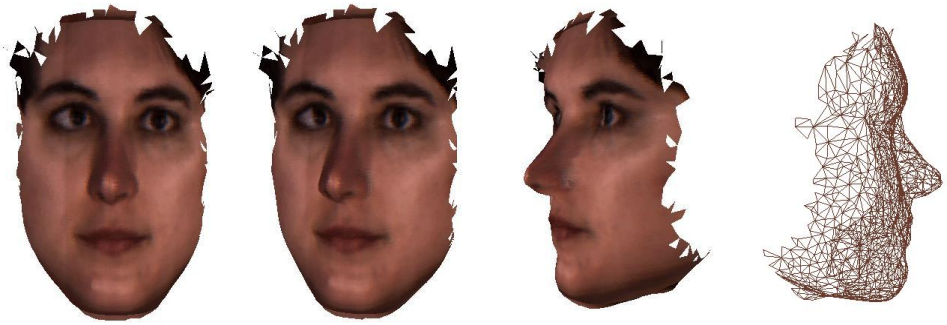

Figure 7.13: Reconstructed model of the first example.

Figure 7.14: Reconstructed model of the second example.

7.3 Discussion

The deformation method represented in this chapter is based on the assumption that the generic mesh is an average geometrical representation of a human face. The threshold values therefore are adjusted so that large deviations from this model are rejected. However, since the generic mesh gives the face model of a male white person, this assumption may not be true in some cases. Among these cases we may mention faces belonging to African black male and females, young children (below 5 years old), Asian faces, and some female face. To show the effect of our assumption in the above mentioned cases, the 3D face model of an Asian child is represented. Figure 7.15 shows the original image. The

Figure 7.15: Original image of an Asian child.

reconstructed model of the Asian child from different views is given in Figure 7.16. The main problems in creating the 3D model of an Asian child can be listed as below:

1. The forehead in a child is much more wider than an adult. This fact intensifies the local deformation step and causes other facial features to get far from the expected average value and therefore rejected during deformation step. The forehead regions of a child and an adult are compared in Figure 7.17.

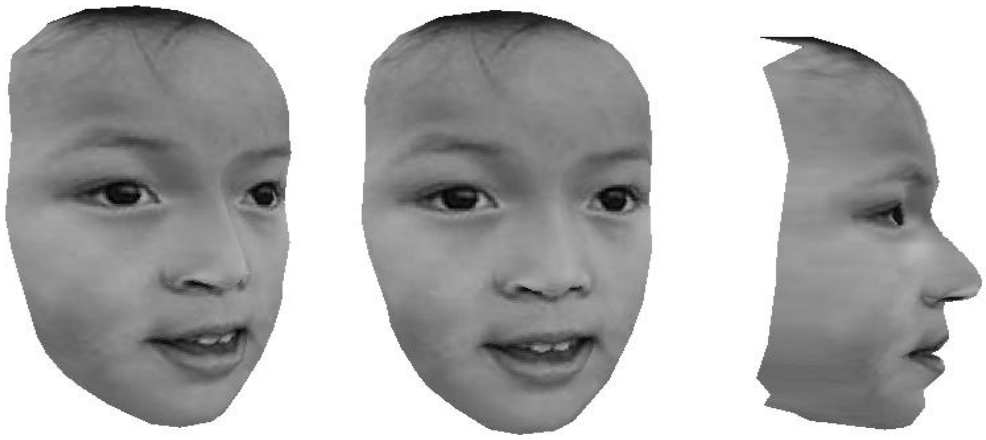

Figure 7.16: 3D model of an Asian child.

2. Nose is shorter and wider in a child than an adult 7.17.

3. In Asian people, due to the different form of the eyes, the distance between eyes and eyebrows is wider.

4. Lower half of the head is thinner than the upper half in a child. This means the head is quite large in a child compared to an adult.

The effect of wide eyelids is clear from reconstructed models 7.16. In this example we have also a smile in the face of the child. The profile image of the child shows that the deformation of the lips are not as much as we expect from the frontal views.

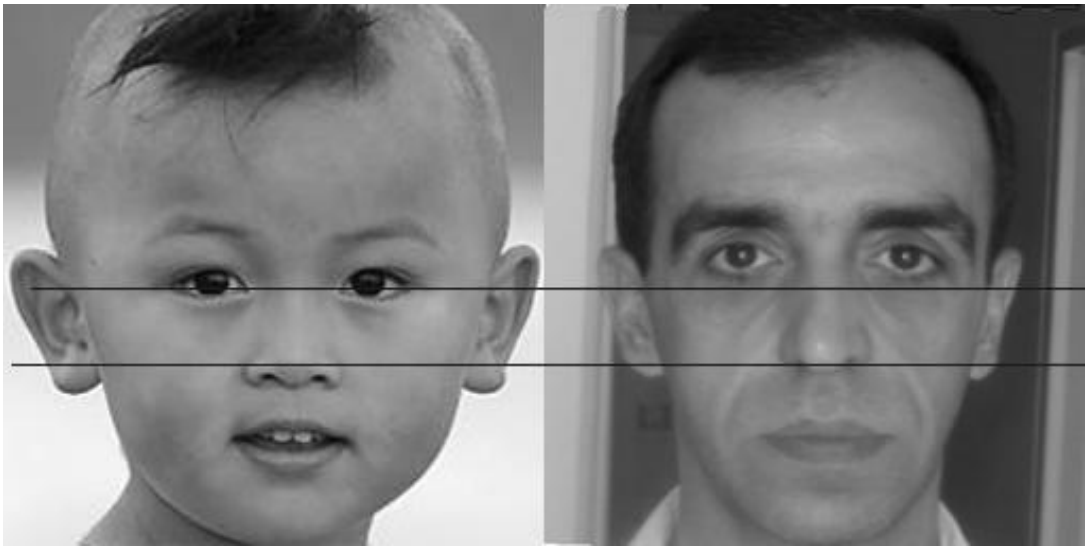

Figure 7.17: Comparison a child and an adult facial areas.

CHAPTER 8

Conclusion and Future Work

In this thesis, we describe a method for 3D human face modeling using uncalibrated images. Having uncalibrated images as input makes the method generic and applicable for any system setup on one hand but introduces the degeneracy problem in estimating internal camera parameters and camera motions on the other hand. We describe a solution to the degeneracy problem in camera calibration for the most common case. This case of degeneracy occurs when the camera motion is about a single axis or parallel axes. The single axis degeneracy in image acquisition happens very frequently, because the most important information about the texture and geometry of human head in provided by front and side views. The second main issue in describing geometry of human faces is complexity of the shape and existence of poor texture on the facial skin. This makes finding sufficiently many point matches and their reliable reconstruction difficult. Our solution for the reliable reconstruction of facial geometry is based on a generic model. This model may be considered as an average geometric model of a large number of faces and therefore not far from any of them. In this thesis we use a generic model for all cases however, a more reliable and accurate geometric approximation for a face can be found if the anthropological features

of human faces are considered. Different races show different anthropological and physical properties. Generic model which averages over the examples from a given race or nation could provide a better approximation. Same argument is valid when we consider gender and age. This better geometric estimation has the cost of race, gender or age estimation from images during reconstruction phase to choose the most appropriate model.

Generic model differs from any specific face model slightly. To obtain the geometry of any particular face we have to deform the generic model. The deformation should be restricted to a limited local area to make them manageable. On the other hand the symmetry existing in human faces and anthropologic relationship between facial features create dependency between facial feature deformations. However, relating deformation in facial features to each other decreases geometric accuracy of the model. To propagate the movement of a vertex to its neighboring vertices in a controlled manner, we consider the links connecting vertices to be massless springs reacting to forces applied to them. The springs are in the equilibrium state at the beginning and after any change in the location of a vertex they gain the equilibrium state again. We also apply the well known and mathematically sound Delaunay meshing algorithm to human face modeling. This application makes possible to increase mesh density locally by inserting new vertices and keeping the consistency in the model. Our method has the following features not available in other methods.

- Camera calibration and 3D face model generation are integrated in a single algorithm.
- We are not considering any specific polygonal mesh and the algorithm can be implemented with different order of complexity in meshes.
- Degenerate motions do not deteriorate the system.
- The amount of reliability of the algorithm due to invalid assumptions in camera parameters are predictable.

We have introduces a solution for the degeneracy in auto calibration when the camera rotation is about a single axis or parallel axes. In the similar systems, either camera calibration is done before modeling [20], or no calibration is carried our and the model is approximated by bundle adjustment [9]. We have described an experimental and comparative study of the sensitivity of reconstruction and auto calibration algorithms to improper assumptions in intrinsic camera parameters which has not been studied in the literature. We also describe an algorithm for mesh deformation using a cloud of 3D points and local refinements. In this algorithm, we apply Delaunay triangulation method to local enhancements of human face model. Applying Delaunay algorithm in 3D causes some degeneracies. We describe an algorithm for solving degeneracy in applying Delaunay triangulation to 3D meshes. We have also developed an integrated system for modeling human faces from uncalibrated images. In our modeling method we use a generic mesh which has also the benefit of fast rendering and easy establishing facial expressions in animation. Generally facial expressions such as smiling, anger, fear, etc are added to the model by displacing some of the vertices which are important in displaying these expressions. To add the effect of these displacements or deformations on the other parts of the face, facial muscles are simulated. These muscles can be implicitly given by the coefficients of the springs connecting vertices to each other in the generic mesh. This means the model generated using spring based mesh is suitable for animation applications.

Another application for the 3D face models generated using this method is pose independent identification. Most of the systems developed for identification are based on image acquisition from a predefined view (generally front view). These systems fail to identify a person when the image in taken from a different view. A solution for this problem is establishing a database of 3D models and identifying the persons by rotating the model to the same pose and performing the comparison.

REFERENCES

[1] Faugeras, O.D. and Luong, Q.T. and Papadopoulo, T.,2001,The Geometry of Multiple Images, MIT Press.

[2] Hartley, R.I. and Zisserman, A., 2000, Multiple View Geometry in Computer Vision, Cambridge Press.

[3] Fenn Roger, 2000, Geometry, Springer press.

[4] Faugeras O., 1992. What can be seen in three dimensions with an uncalibrated stereo rig, European conference Computer Vision, pp. 563-578.

[5] Faugeras, O. 1995, Stratification of 3-D vision: projective, affine, and metric representations, Journal of the Optical Society of America A12(3), pp. 465-484.

[6] Faugeras O., Luong Q. , Maybank S., 1992. Camera self-calibration: Theory and experiments, In proc. European Conference on Computer Vision, LNCS 588, pp. 321-334. Springer-Verlag.

[7] Faugeras, O., Robert, L.: 1994, What can two images tell us about a third one?, in Proceedings of the 3rd European Conference on Computer Vision, pp. 485-492.

[8] Fletcher, R., 1980. Practical Methods of Optimization, Vol. 1, Unconstrained Optimization, and Vol. 2, Constrained Optimization, John Wiley and Sons.

[9] Fua, P., 2000, Regularized bundle-adjustment to model heads from image sequences without calibration data, International Journal of Computer Vision, pp. 153-171.

[10] Hartley, R. 1995, In defense of the 8-point algorithm, Proceedings of the 5th International Conference on Computer Vision, IEEE Computer Society Press, Boston, MA, pp. 1064-1070.

[11] Hassanpour R. and Atalay V., 2002. Head Modeling with Camera Autocalibration and Deformation, 7th Int. Fall Workshop on Vision, Modeling and Visualization-VMV 2002, Erlangen, Germany, pp.3-11

[12] Hassanpour R., Atalay V., 2002. Camera Auto-Calibration using a Sequence of 2D Images with Small Rotations and Translations, Proceedings of the 17th Int. Symposium on Computer and Information Sciences, Orlando, Florida, USA, pp.215-219, CRC Press.

[13] Lengagne, R, Fua, P, Monga, O, 1998. 3D face modeling from stereo and differential constraints, International Conference on Pattern Recognition, pp. 148-153.

[14] Otlu B., Hassanpour R., Atalay V., 2002. Consistency of projective reconstructions from an image pair, 9th International Workshop on Systems, Signals and Image Processing, Manchester Town Hall, UK. pp. 376-381.

[15] Pollefeys, M., Van Gool L., Oosterlinck A., 1996. The Modulus Constraint: a new constraint for self-calibration, In Proc. International Conference on Pattern Recognition, pp. 31-42.

[16] Triggs, B. and McLauchlan, P. and Hartley, R. and Fitzgibbon, A., 2000. Bundle Adjustment: A Modern Synthesis, Vision Algorithms: Theory and Practice volume 1883, pp. 298-372.

[17] Triggs, B., 1997. Autocalibration and the absolute quadric, In Proceedings of the Conference on Computer Vision and Pattern Recognition, Puerto Rico, USA, pp. 607-614.

[18] William H. Press, Numerical recipes in C : the art of scientific computing , 2nd ed. Cambridge University Press , 1992.

[19] Zeller C., 1996. Projective, Affine and Euclidean Calibration in Computer Vision and Application of Three Dimensional Perception, PhD thesis, Robot Vis Group, INRIA Sophia-Antipolis.

[20] Zhang Z. 2001. Image-based Modeling of Objects and Human Faces, Proceedings of SPIE Vol.4309 , Videometrics and Optical Methods for 3D Shape Measurement , pp. 1-15.

[21] Zhang Z., 1996. Determining the Epipolar Geometry and its Uncertainty: A Review, Technical Report 2927, INRIA, Sophia Antipolis, 12.

[22] Zhang, Z. and Isono, K. and Akamatsu, S.,1998, Euclidean Structure from Uncalibrated Images Using Fuzzy Domain Knowledge: Application to Facial Images Synthesis, International Conference on Computer Vision, pp. 784-789.

[23] Agapito L., Hayman E., and Reid I.D., 2001, Self Calibration of Rotating and Zooming Cameras, International Journal of Computer Vision, Vol.45, No.2, pp.107-127.

[24] Armstrong M., Zisserman A., and Hartley R., 1996, Self-calibration from image triplets, European Conference on Computer Vision, LNCS 1064/5, pp.3-16. O.

[25] Faugeras, Q. Luong, "Self-calibration of a stereo rig from unknown camera motions and point correspondences", Research Report Nr. 2012, INRIA Sophia Antipolis

[26] Faugeras O.D, Quan L, and Sturm P., 2000, Self-calibration of a 1D projective camera and its application to the self-calibration of a 2D projective camera, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.22, No.10, pp.1179- 1185.

[27] Fusiello A., 2001, A New Autocalibration Algorithm: Experimental Evaluation, Int. Conference on Computer Analysis of Images and Patterns, LNCS 2124, pp.717-725.

[28] Fusiello A., 2000, Uncalibrated Euclidean reconstruction: A review, Image and Vision Computing, Vol.18, No.67, pp.555-563.

[29] Hartley, R.I., and Kaucic R., 2002, Sensitivity of Calibration to Principal Point Position, European Conference on Computer Vision, LNCS 2351, pp.433-542.

[30] Hartley R.I., 1992, Estimation of relative camera positions for uncalibrated cameras, European Conference on Computer Vision, LNCS 588, pp.579587.

[31] Hartley, R.I., 1992, Calibration of Cameras Using the Essential Matrix, DARPA Image Understanding Workshop, pp.911-915.

[32] Kahl F. and Triggs B., 1999, Critical motions in euclidean structure from motion, Int. Conference on Computer Vision and Pattern Recognition, pp.366-372.

[33] Kahl F., Triggs B., and Sturm K., 2000, Critical motions for autocalibration when some intrinsic parameters can vary, Journal of Mathematical Imaging and Vision, Vol.13, No.2, pp.131-146.

[34] Lengagne, R., Fua, P., and Monga, O., 1998, 3D face modeling from stereo and differential constraints, International Conference on Pattern Recognition, pp.148- 153.

[35] Mendonca P., and Cipolla R., 1999, A simple technique for self-calibration, Int. Conference on Computer Vision and Pattern Recognition, pp.500-505.

[36] Oliensis J., 1996, Structure from linear or planar motions, Int. Conference on Computer Vision and Pattern Recognition, pp.335-342.

[37] Pollefeys M., Koch R., and Van Gool L., 1998, Self calibration and metric reconstruction in spite of varying and unknown internal camera parameters, International Conference on Computer Vision, pp.90-96.

[38] Torr P.H.S., Fitzgibbon, A.W. and Zisserman, A., 1999, The Problem of Degeneracy in Structure and Motion Recovery from Uncalibrated Image Sequences, International Journal of Computer Vision, Vol.32, No.1 pp.27-44.

[39] Triggs B, 1998, Autocalibration from Planar Scenes, LNCS 1406, pp.89-109.

[40] Zhang, Z., Isono, K. and Akamatsu, S., 1998, "Euclidean Structure from Uncalibrated Images Using Fuzzy Domain Knowledge: Application to Facial Images Synthesis", International Conference on Computer Vision, pp.784-789.

[41] Heyden A., and Astrom. K, 1997, Euclidean reconstruction from image sequences with varying and unknown focal length and principal point. In CVPR, pages 438-443.

[42] Akimoto T. , Suenago V., 1993, Automatic creation of 3D facial models, IEEE trans, Computer Graphics and application vol. 3 pp 16-22

[43] Luo S., King W. R., 1996, Automatic Human Face Modeling in Model-Based Facial Image Coding, Proc. of AUstralian New Zealnd Conf. on Intelligent Information Systems, pp 167-171.

[44] Yan J. , Gao W. , 1998 , Deformable Model-Based Generation of Realistic 3-D Specific Human Face, Proceedings of ICSP, pp 857-860.

[45] Blanz V., Vetter T., 1999 A morohable model for the synthesis of 3D faces, In computer Graphics, Annual Conference Series,SIGGRAPH, pp 187-194

[46] Thalman N., Minh H., Angelis M. and Thalman D., 1989, Design transformation and animation of human faces, Visual Computer (5):32-39.

[47] Parke F., 1996, Computer Facial Animation, AKPeters, Wellesley, Massachusetts.

[48] Gallier J., 2000 , Geometric Models and Applications, Spronger press.

[49] Harris C. Stephens M. , 1988 , A combined corner and edge detector, In proc of 4th Alvey Vision Conf, pp 189-192.

[50] Pighin F. Hecker J., 1998 , Synthesizing realistic facial expressions from photographs, Computer Graphics, Annual conference Series, pp 75-84.

VITA

Reza Zare Hassanpour was born in Orumiyeh Iran on May 24, 1963. He started his study in computer hardware enginering at the Department of Computer Engineering, Shiraz University, Shiraz-Iran in 1991. He graduated in 1995 with honors and his senior project entitled "Development of the kernel part of a multiprocess operating system for 80386 and 80486 machines" was awarded as the best project. In 1995, he commenced his master's study at the Polytechnic University of Tehran in robotics group of Computer Engineering Department where he also started to work as a system analyzer and programmer in the research and development center of the university. He received his M.Sc. degree in 1998. The subject of his M.Sc. Thesis was "Human face detection in cluttered images". In 1998 he commenced his Ph.D. study at the image processing and computer vision group in the Department of Computer Engineering, Middle East Technical University, Ankara-Turkey under the supervision of Dr. Volkan Atalay. He has been an instructor in Azad University in Iran from 1996 to 1998 and Çankaya University, Ankara-Turkey since 1999. His main interest areas are 3D modeling, Analyzing and modeling human activities, man-machine interaction.