# Ανάπτυξη διαδραστικής εφαρμογής ανακατασκευής 3Δ σκηνής από φωτογραφίες

195

ΜΠΛΕ

Η ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ ΕΞΕΙΔΙΚΕΥΣΗΣ

υποβάλλεται στην

ορισθείσα από την Γενική Συνέλευση Ειδικής Σύνθεσης

του Τμήματος Πληροφορικής Εξεταστική Επιτροπή

από τον

Χαρίλαο Καλογήρου

ως μέρος των Υποχρεώσεων για τη λήψη του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΔΙΠΛΩΜΑΤΟΣ ΣΤΗΝ ΠΛΗΡΟΦΟΡΙΚΗ

ΜΕ ΕΞΕΙΔΙΚΕΥΣΗ

ΣΤΟ ΛΟΓΙΣΜΙΚΟ

Ιανουάριος 2006

# ACKNOWLEDGMENTS

I would like to thank my supervisor Professor Ioannis Fudos for his help, support, and the patience that he has shown during the elaboration of this thesis.

# Contents

# List of Figures

# List of Tables

# List of Symbols

| | | |
|---|---|---|
| $\mathbf{l}$ | — | A line |
| $\tilde{\mathbf{x}}$ | — | Homogeneous coordinates vector of vector $\mathbf{x}$ |
| $\mathbf{X}$ | — | World space point |
| $\mathbf{K}$ | — | Camera calibration matrix |
| $\mathbf{P}$ | — | Camera projection matrix |
| $\mathbf{R}$ | — | Rotation matrix |
| $\mathbf{t}$ | — | Translation matrix |
| $\mathbf{F}$ | — | Fundamental matrix |
| $\mathbf{E}$ | — | Essential matrix |
| $\omega_\infty$ | — | Image of the absolute conic |
| $\mathbf{A}^\mathsf{T}$ | — | The transpose of $\mathbf{A}$ |
| $\mathbf{A}^{-1}$ | — | The inverse of $\mathbf{A}$ |

# ΠΕΡΙΛΗΨΗ

Χαρίλαος Καλογήρου του Κωνσταντίνου και της Ασπασίας.
MSc, Τμήμα Πληροφορικής, Πανεπιστήμιο Ιωαννίνων, Γενάρης 2006.
Ανάπτυξη διαδραστικής εφαρμογής ανακατασκευής 3Δ σκηνής από φωτογραφίες
Επιβλέπων : Ιωάννης Φούντος

Στην εργασία αυτή μελετάμε το πρόβλημα της ανακατασκευής σκηνών τριών διαστάσεων από φωτογραφίες που έχουν ληφθεί με φωτογραφικές μηχανές αγνώστων παραμέτρων, εστιάζοντας στις λεπτομέρειες της υλοποίησης και στα προβλήματα που παρουσιάζονται. Ο στόχος είναι να παρέχουμε ένα διαδραστικό πλαίσιο που θα επιτρέπει στον χρήστη να ανακατασκευάζει επαρκώς τρισδιάστατες σκηνές από φωτογραφίες. Το πρόβλημα της ανακατασκευής μελετήθηκε από διάφορες οπτικές γωνίες και αξιολογήσαμε την απόδοση των διαφόρων προτεινόμενων μεθόδων. Το τελικό πλαίσιο ανακατασκευής αποτελείται από τρία βήματα. Το πρώτο βήμα περιλαμβάνει την βαθμονόμηση των φωτογραφικών μηχανών, με την εξάγουμε τις εσωτερικές παραμέτρους των φωτογραφικών μηχανών. Στην συνέχεια το δεύτερο βήμα αποτελείται από το ταίριασμα χαρακτηριστικών σημείων πάνω στις φωτογραφίες. Τα χαρακτηριστικά αυτά σημεία χρησιμοποιούνται στην συνέχεια για τον υπολογισμό της epipolar γεωμετρίας μεταξύ των διαθέσιμων φωτογραφιών. Τέλος αφού έχουμε υπολογίσει την epipolar γεωμετρία των βαθμονομημένων φωτογραφιών, συνεχίζουμε με την ανάκτηση των εξωτερικών παραμέτρων των φωτογραφικών μηχανών, δηλαδή την θέση τους στον χώρο, το οποίο θα μας επιτρέψει να αναχτήσουμε το τρισδιάστατο μοντέλο της σκηνής μέσο τριγωνοποίησης. Η εφαρμογή που αναπτύχθηκε είναι ένα αλληλεπιδραστικό σύστημα που επιτρέπει στον χρήστη να εισάγει φωτογραφίες από διάφορους τύπους αρχείων εικόνας και να εφαρμόσει τα παραπάνω βήματα ώστε να ανακατασκευάσει την σκηνή. Το τρισδιάστατο μοντέλο της σκηνής μπορεί στην συνέχεια να εξαχθεί σε άλλα εργαλεία για περαιτέρω επεξεργασία.

# Abstract

Charilaos Kalogirou.
MSc, Computer Science Department, University of Ioannina, Greece. January 2006
Supervisor : Ioannis Fudos

In this thesis we study the problem of reconstructing three-dimensional scenes from two-dimensional snapshots acquired from uncalibrated cameras. We focus on the implementation details and problems encountered. The goal is to supply an interactive framework that will enable the user to efficiently reconstruct three-dimensional scenes from photographs. The reconstruction problem is studied from various perspectives and we evaluate the efficiency of the proposed methods. The proposed reconstruction framework consists of three steps. The first step consists of calibrating the cameras, i.e. we obtain the intrinsic parameters of the cameras. Then the second step consists of feature point matching on photographs. These feature points are then used to estimate the epipolar geometry among the available views. Finally, once the epipolar geometry of the calibrated views is known, we derive the positioning of the cameras in world space coordinate system, i.e. obtain a three-dimensional model of the scene with triangulation. We have developed an interactive system that enables the user to import two-dimensional snapshots in any digital image format and to perform the above mentioned steps to recreate the scene. The three-dimensional model can then be exported to solid modelling or reverse engineering software for further processing and editing.

# Chapter 1

# Introduction

## 1.1 Overview

The objective of this thesis is to explore, present and implement three-dimensional reconstruction techniques from images. We will implement a system capable of acquiring three-dimensional models directly from two dimensional snapshots of a real object.

The problem of acquiring cloud points from images is a fairly new research field. Researchers have been investigating methods for acquiring three-dimensional information from images for many years, mainly for use in robot vision where a crude approximation of the environment is sufficient. That has changed recently with the increasing demand for realistic three-dimensional models for use in CAD, cinema, virtual reality, computer games and other computer graphics applications. The capabilities of personal computers today make it feasible to render high resolution and life like 3D models. Three-dimensional modelling of real world models is a time consuming and therefore expensive process. Thus creating accurate three-dimensional models from real objects without the use of expensive hardware is essential to Computer Graphics.

In this thesis we present various methods that has been proposed to solve the problem, along with the complexity and performance evaluation of their implementations.

## 1.2 Definition of the problem

In a more elaborate definition, the problem we are trying to solve involves acquiring three dimensional characteristics of objects when we have available images *(photographs)* taken with off-the-shelf consumer cameras.

The problem's nature is best summarized by observing that images are the product of an irreversible projection operation from a three-dimensional scene onto a two-dimensional image. During this operation the depth information is lost. This is illustrated in Figure 1.1. Notice that the projected point on the image can be anywhere on the line of sight.

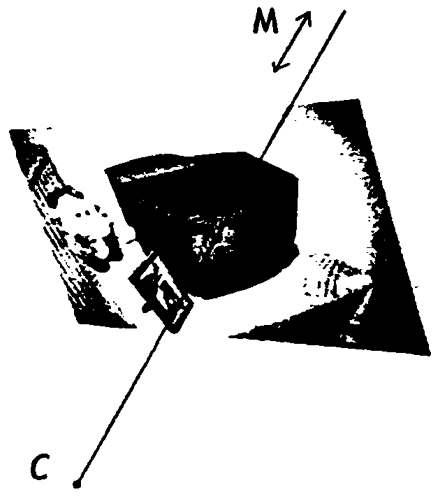


Figure 1.1: The back projected point can be anywhere along the line of sight

Therefore the information an image provides is not sufficient for three-dimensional reconstruction. We must develop algorithms that overcome the projection's ambiguity, possibly by using information from two of more images.

It turns out that the use of two or more images can resolve the depth ambiguity. The algorithm to do so is well known in topography and map making. Suppose that we have two images of a scene and two corresponding points, one on each image, projections of a three-dimensional point in the scene. The actual three-dimensional point can be acquired from the intersection of two lines of sight as presented at Figure 1.2. This process is known as triangulation.

To use triangulation we should acquire the following information:

- Corresponding image points
- The parameters of the cameras

In this thesis we examine how we can acquire the parameters of the cameras from corresponding image points. These parameters are called the *motion* of the camera. We then use this information to derive the three-dimensional *structure* of the scene .

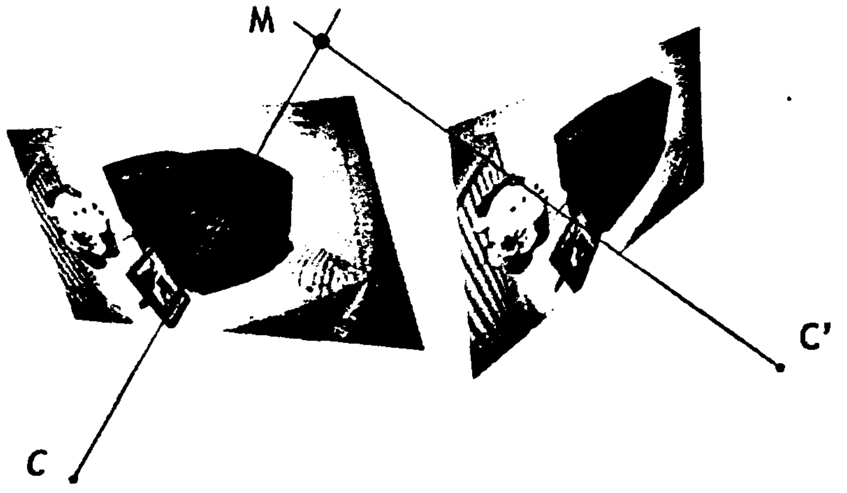


Figure 1.2: Acquiring three-dimensional point with triangulation

The rest of this thesis is organized as follows. Chapter 2 discusses the fundamentals of image formation. We derive a model for the camera that we will use throughout our work. In Chapter 3 we present a first approach to the problem. We study the proposed method and we determine the scope and limitations based on the parameters of the problem we are trying to solve. In Chapter 4 we present a more advanced and efficient method for three-dimensional reconstruction. We present the constraints of two-view geometry and how we can exploit it to estimate both motion and structure. Chapter 5 discusses the development of the three-dimensional reconstruction framework. The discussion ranges from the choice of programming language, to the implementation methods and systems used to efficiently implement the theoretical algorithms presented in previous chapters. Finally Chapter 6 concludes this thesis with a summary of our contribution and identification of future research direction.

# CHAPTER 2

# MODELLING THE IMAGE FORMATION PROCESS

## 2.1 Introduction

Before we explore the possible solutions to the problem of reconstructing a scene from a set of images, we should first understand the process of image formation. We should have a sound mathematical model of the workings that describe the creation of a photograph in a camera. A sound model does not necessarily mean a physically accurate model. We will design a model that is suitable for our purposes while being constrained on complexity in order to be efficient. For the purposes of our system we require a simple geometric model of the image formation process, and not require more complex photometric models. We use common theory for lenses that originate in physics and we formulate a model of a camera that successfully approximates the functionality of a real camera. In this chapter we derive the model that will be used in our work, and we explain why it is suitable for our needs.

## 2.2 Representing images

An image at an abstraction level, is a two dimensional brightness array. In the case of a standard camera, an image is a map $I$, defined on a planar and rectangular two dimensional surface $\Omega$, taking positive real numbers. Therefore $I$ is a function:

$$I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}_+ ; (x, y) \mapsto I(x, y) \tag{2.1}$$

In the case of digital images both $\Omega$ and $\mathbb{R}_+$ are discretized. For example $\Omega$ can be $[1, 640] \times [1, 480] \subset \mathbb{Z}^2$, and $\mathbb{R}_+$ can be approximated by an interval of integers $[0, 255] \subset \mathbb{Z}_+$. The values of the image $I$ can then be presented on the computer's monitor by using the values as intensities for pixels.

## 2.3 Light and lenses

To complete the image formulation process, we must describe how the values of $I(x, y)$ at each point $(x, y)$ on $\Omega$ are calculated. The main component of a camera is the set of lenses. These lenses are used to direct light on the photographic film, or sensor in the case of a digital camera. It is true that completely modelling the way lenses interact with light can be extremely complex. In physics scientists try to model this effect by assuming "special" forms of lenses. Next we will review two basic lens models.

### 2.3.1 Thin lens

A basic mathematical model to describe and study the way lenses refract light is the *thin lens*(Figure 2.1). This mathematical model is described by an axis, called the *optical axis*, and a plane perpendicular to the axis, called the *focal plane*. The intersection of the focal plane with the optical axis is called the *optical center*. The thin lens has two parameters. The *focal length* $f$ and its diameter $d$. Its operation is described by two properties:

- All rays entering the lens parallel to the optical axis intersect on the optical axis at a distance $f$ from the optical center. The point of intersection is called the focus of the lens.

- All rays through the optical center are undeflected.

To make it clear lets try an example. Suppose that point $p \in \mathbb{E}^2$ is at distance $Z$ from the optical center along the optical axis. We then draw two rays starting from point $p$, one parallel to the optical axis and one through the optical center. According to the properties of thin lens, the first ray intersects with the optical axis at the focus point, while the second passes through the lens undeflected. Call $x$ the point where the

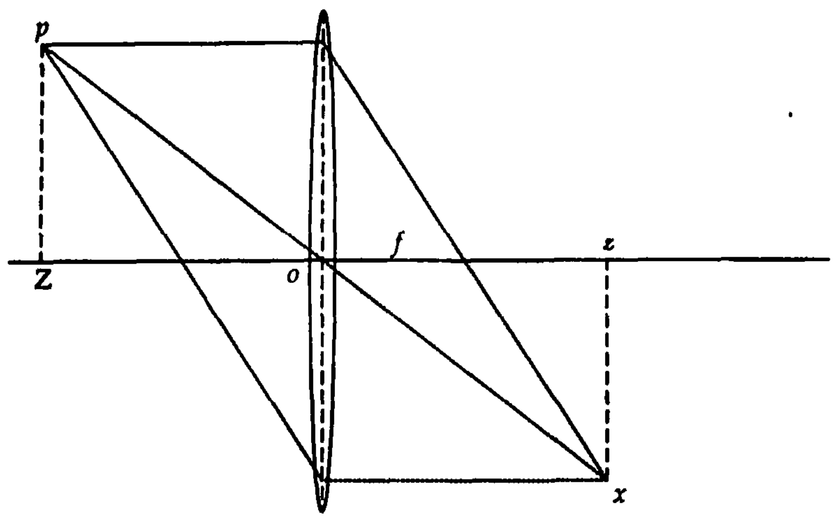


Figure 2.1: The thin lens model

rays intersect, and let $z$ be the distance of $x$ from the optical center along the optical axis. The above rays can be viewed in Figure 2.1. Using similar triangles in Figure 2.1 we obtain the following *fundamental equation of the thin lens*:

$$\frac{1}{Z} + \frac{1}{z} = \frac{1}{f} \tag{2.2}$$

The point $x$ is called the "image" of point $p$. Therefore, under the assumption of thin lens, $I(x)$ at the point $x$ with coordinates $(x, y)$ on the image plane, or retinal plane, is obtained by integrating all the energy emitted from a region of space contained in the cone determined by the geometry of the lens.

### 2.3.2 Pinhole lens

Suppose now that the aperture (radius) of a thin lens decrease to zero. Then all rays are forced to go through the optical center $o$. So all rays remain undeflected. Consequently the aperture of the cone decreases to zero and the only points that contribute to $I(x, y)$ are the points on a line through the center $o$ of the lens.

Suppose that we have a reference frame centered at $o$ with its $z$ axis being the optical axis, and let point $p$ have coordinates $\mathbf{X} = [X, Y, Z]^\top$. Then with similar triangles again in Figure 2.2 we find out that the coordinates of $p$ and its image $x$ are related by the well known *perspective projection*:

$$x = -f\frac{X}{Z}, \quad y = -f\frac{Y}{Z} \tag{2.3}$$

where $f$ is the *focal length*. This imaging model is also called the *ideal pinhole camera model.* Note that the minus sign in the equations result in the image being presented upside-down on the retinal plane. We can remove the negative sign by flipping the

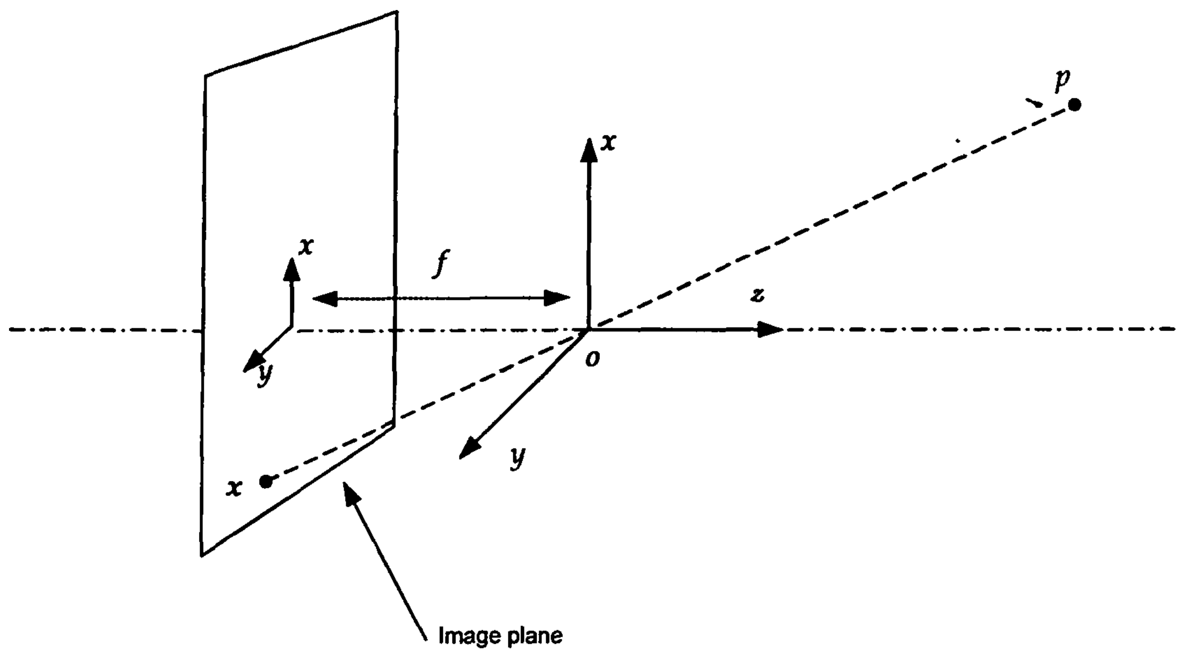


Figure 2.2: The pinhole lens model

image space coordinates or by moving the retinal plane in front of the optical center.

## 2.4 Geometric model for image formation

So now under the assumption of a pinhole camera, we can essentially reduce the image formation process to tracing rays from points on objects to pixels on the retinal plane. To establish a precise correspondence between points in three-dimensional space with their projected images on the retinal plane, a mathematical model for this process must account for three types of transformations:

- Coordinate transformation between the camera frame and the world frame
- projection of 3D coordinates onto 2D image coordinates
- coordinate transformation between possible choices of image coordinate frames

The coordinate transformation between the camera frame and the world frame, essentially specifies the position of the camera in the scene, so it is a rigid body transformation based on the position of the camera. The projection transformation is where the actual irreversible transformation takes place. This transformation projects the three-dimensional space on a two dimensional plane. The last transformation takes place on the two dimensional plane that the image is formulated and has to do with the choice of coordinate frame for the two-dimensional snapshots. All the above transformations will be discussed further in this chapter and their functionality will be become more clear when we specifically define the camera model that will be used in this thesis.

### 2.4.1 The ideal perspective camera

Consider a point $p$ with world space coordinates $\mathbf{X}_0 = [X_0, Y_0, Z_0]^\top$. Then the coordinates $\mathbf{X} = [X, Y, Z]^\top$ of the same point relative to the camera space coordinates are given by a rigid-body transformation $g = (\mathbf{R}, \mathbf{T})$ of $\mathbf{X}_0$:

$$\mathbf{X} = \mathbf{R}\mathbf{X}_0 + \mathbf{T} \tag{2.4}$$

Suppose now that we use the pinhole camera model. The point $\mathbf{X}$ is then projected onto the retinal plane at the point:

$$\mathbf{x} = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z}\begin{bmatrix} X \\ Y \end{bmatrix}$$

We can rewrite this relationship in homogeneous coordinates like this:

$$Z\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.5}$$

or if we specify $\mathbf{x}, \mathbf{X}$ using homogeneous coordinates:

$$Z\mathbf{x} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}\mathbf{X} \tag{2.6}$$

In the above equation we can decompose the matrix to:

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.7}$$

We call the first matrix $\mathbf{K}_f$ and the second $\mathbf{\Pi}_0$. The matrix $\mathbf{\Pi}_0$ in referred as the *canonical projection matrix*. From the world frame to camera frame transformation we have:

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0}_3^\top & 1 \end{bmatrix}\begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{2.8}$$

Summarizing all the above we get the overall geometric model for an ideal camera:

$$Z\mathbf{x} = \mathbf{K}_f\mathbf{\Pi}_0\mathbf{X} = \mathbf{K}_f\mathbf{\Pi}_0 g\mathbf{X}_0 \tag{2.9}$$

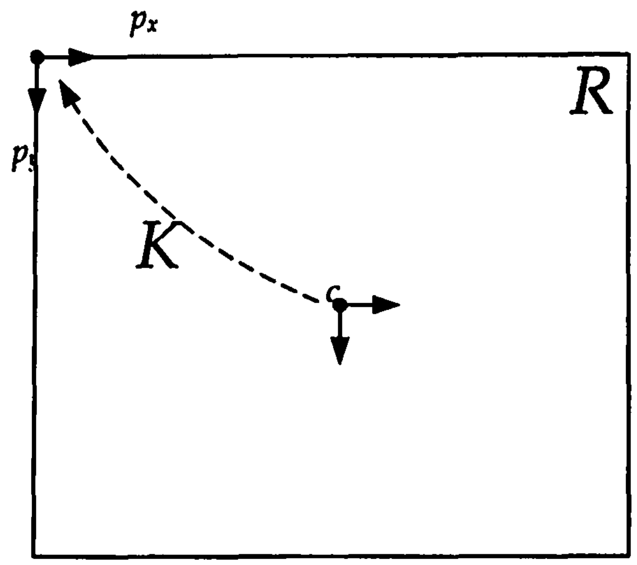


Figure 2.3: From retinal coordinates to pixel coordinates

In the case the focal length is known and hence can be normalized to 1, this model reduces to an Euclidian transformation $g$ followed by a canonical projection $\Pi_0$.

### 2.4.2 A camera with intrinsic parameters

The ideal camera model uses a very specific retinal frame. The retinal frame is centered at the optical center with one axis aligned with the optical axis. When dealing with images on computers however the retinal coordinate frame is usually different. The coordinates are referenced in pixels.with origin of the image being the upper-left corner of the image. So to use the model described by Equation (2.9), we need to specify the transformation from the retinal frame to the pixel array as shown in Figure 2.3

The first transformation to be applied is a scale on the retinal coordinates so that they map to pixel coordinates. This transformation is described by a *scaling matrix*:

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} \frac{1}{p_x} & 0 \\ 0 & \frac{1}{p_y} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{2.10}$$

However the scaled coordinates are still specified relative to the *principal point*. This point is the point where the $z$-axis intersects with retinal plane. So we need to translate the origin of the reference frame to the upper-left corner. To do this we add $c_x$, $c_y$ to $x_s$ and $y_s$. Where $(c_x, c_y)$ are the coordinates of the principal point relative to the reference frame. So if the actual image coordinates are given by the vector $\mathbf{x}' = [x', y', 1]^\top$ instead of the ideal image coordinates $\mathbf{x} = [x, y, 1]^\top$ then the coordinate transformation can be written as:

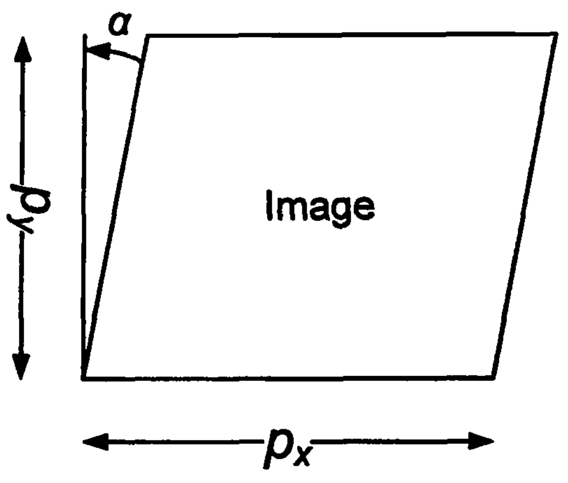


Figure 2.4: Image skew

$$\mathbf{x}' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{p_x} & 0 & c_x \\ 0 & \frac{1}{p_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{2.11}$$

In the case where the pixels are not rectangular a more general scaling matrix can be used that takes that into account:

$$\begin{bmatrix} \frac{1}{p_x} & \frac{1}{p_y}\tan a \\ 0 & \frac{1}{p_y} \end{bmatrix} \tag{2.12}$$

where $a$ is the angle of the pixel's borders as viewed in Figure 2.4. With most cameras it is common to assume that the skew angle is zero. However there are cases that we need to specify an $a$. This is usually on scanned images that where previously taken with regular cameras that use photographic film.

Now if we combine the projection model of the ideal camera with this scaling we gain a more realistic model for cameras:

$$Z \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{p_x} & \frac{1}{p_y}\tan a & c_x \\ 0 & \frac{1}{p_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{2.13}$$

In the above equation we denote the scaling matrix with $\mathbf{K}_s$ and we define $\mathbf{K}$ as:

$$\mathbf{K} = \mathbf{K}_s\mathbf{K}_f = \begin{bmatrix} f\frac{1}{p_x} & f\frac{1}{p_y}\tan a & c_x \\ 0 & f\frac{1}{p_y} & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

then the final model for a camera with intrinsic parameters is:

$$Z\mathbf{x}' = \mathbf{K}\Pi_0\mathbf{X} \tag{2.15}$$

The upper triangular matrix $\mathbf{K}$ collects all the parameters that are *intrinsic* to the camera and so we call this matrix the *intrinsic parameter matrix*, or the *calibration matrix* of the camera.

The process of obtaining the camera's calibration matrix is known as *camera calibration*. The camera calibration step allows us to find the matrix $\mathbf{K}$ which can give us normalized coordinates from pixel coordinates with a simple inversion.

# CHAPTER 3

# A FIRST APPROACH TO THREE-DIMENSIONAL RECONSTRUCTION

## 3.1 Camera model

In this first approach the perspective camera model is used, known also as the *pinhole camera model*. The geometric process for image formation with the pinhole camera model is completely determined by specifying a perspective projection center and a retinal plane . The projection of an object point is then obtained as the intersection of a line through that point and the center of projection with the retinal plane. This is illustrated in Figure 3.1, and is basically the *ideal perspective camera* with the retinal plane positioned in front of the camera center.

In this approach we preferred the ideal perspective camera model for its simplicity over the camera with intrinsic parameters model. This way we only have to estimate the focal length of the camera as an intrinsic parameter. The principal point is assumed to be at the center of the two dimensional snapshots, the pixels are rectangular and the skew angles are zero. This close enough to reality for most applications and it was judged a good assumption for the proposed method.

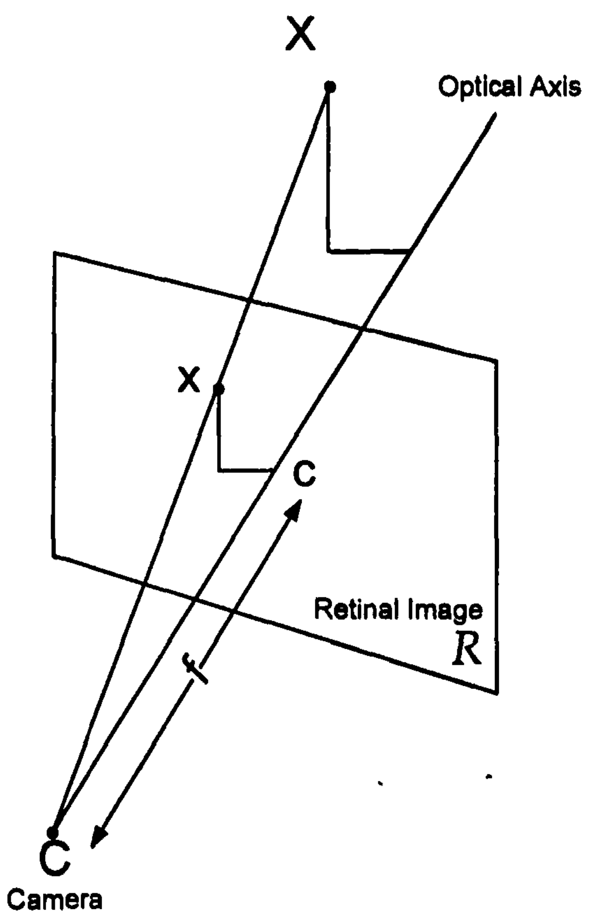


Figure 3.1: Perspective projection

## 3.2 Simultaneous recovery of structure and motion

In this section a method for recovering both camera motion and scene structure will be discussed. The problem will be formulated as an optimization problem that will be solved with the aid of non-linear least squares.

### 3.2.1 Problem formulation

We suppose that we are given $k$ images of a rigid scene and that we want to recover from these images the structure of the scene and the motion of the cameras. This will require to calculate the position and orientation they have in the world frame. The position of three-dimensional points on the surfaces of the scene will have to be calculated also. So we have to calculate for each camera the rotation matrix $\mathbf{R}^k$ and the translation vector $\mathbf{t}^k$, and the three-dimensional points $\mathbf{p}_i$ on the scene's surfaces.

Suppose that the three rows of $\mathbf{R}^k$ are $\mathbf{r}_x^k$, $\mathbf{r}_y^k$ and $\mathbf{r}_z^k$, and the three entries in $\mathbf{t}^k$ are $t_x^k$, $t_y^k$ and $t_z^k$. We also make the assumption that the principal point of the cameras lies at the center of the images, which is common for most cameras. The projection equation (2.9) can be written as:

$$x_i^k = f^k \frac{\mathbf{r}_x^k \mathbf{p}_i + t_x^k}{\mathbf{r}_z^k \mathbf{p}_i + t_z^k} \qquad y_i^k = f^k \frac{\mathbf{r}_y^k \mathbf{p}_i + t_y^k}{\mathbf{r}_z^k \mathbf{p}_i + t_z^k} \tag{3.1}$$

Instead of using the above equation directly, we reformulate the problem to estimate inverse distances to the scene [1]. Let $\eta^k = 1/t_z^k$ be the inverse distance and $s^k = f^k \eta^k$ be the world-to-image scale factor. This formulation allows for the scale factor $s^k$ to be reliably estimated even when the focal length is long, whereas the original formulation has a strong coupling between the $f^k$ and $t_z^k$ parameters. The equations (3.1) become :

$$x_i^k = s^k \frac{\mathbf{r}_x^k \mathbf{p}_i + t_x^k}{1 + \eta^k \mathbf{r}_z^k \mathbf{p}_i} \qquad y_i^k = s^k \frac{\mathbf{r}_y^k \mathbf{p}_i + t_y^k}{1 + \eta^k \mathbf{r}_z^k \mathbf{p}_i} \tag{3.2}$$

finally we collect all the terms in the left side and we get :

$$x_i^k - s^k \frac{\mathbf{r}_x^k \mathbf{p}_i + t_x^k}{1 + \eta^k \mathbf{r}_z^k \mathbf{p}_i} = 0$$

$$y_i^k - s^k \frac{\mathbf{r}_y^k \mathbf{p}_i + t_y^k}{1 + \eta^k \mathbf{r}_z^k \mathbf{p}_i} = 0$$

So we have a system of $2k$ equations that we will solve using the Levenberg-Marquardt non-linear least squares algorithm [1].

### 3.2.2 Simple rotation matrix

The above solution although it is well formulated is has several difficulties in its implementation. The main problem originates in the formulation of the unknown rotation matrix.

The simple approach is to parameterize the $3 \times 3$ rotation matrix with nine unknowns, and try to solve that problem. This turns out not to work efficiently since several geometric properties that the rotation matrix possesses are not satisfied, i.e. the equations constitute a set of necessary but not sufficient conditions. This may result in erroneous solution, unless very good initialization conditions are determined.

Other attempts to enforce the constraints, like including in the optimization process the constraint:

$$\mathbf{R}\mathbf{R}^\top = \mathbf{I}$$

does not give good results because they create singularities that cause the optimization process fail to find the minimum.

### 3.2.3 Euler angles parameterization

In order to enforce the geometric properties we use a specific parameterization of the rotation matrix based on Euler angles .

In $\mathbb{R}^3$ rotations around the $x$, $y$ and $z$ axes are:

$$\mathbf{R}_x(a) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos a & \sin a \\ 0 & -\sin a & \cos a \end{bmatrix}$$

$$\mathbf{R}_y(a) = \begin{bmatrix} \cos a & 0 & -\sin a \\ 0 & 1 & 0 \\ \sin a & 0 & \cos a \end{bmatrix}$$

$$\mathbf{R}_z(a) = \begin{bmatrix} \cos a & \sin a & 0 \\ -\sin a & \cos a & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Any rotation can be given as a composition of rotations about these three axes, according to Euler's rotation theorem. So our rotation matrix is the result of multiplying the above matrices:

$$\mathbf{R}_{all}(a,b,c) = \mathbf{R}_x(a)\ \mathbf{R}_y(b)\ \mathbf{R}_z(c) \tag{3.3}$$

which results in the following parameterization:

$$\mathbf{R}_{all}(a,b,c) = \begin{bmatrix} \cos b \cos c & \cos b \sin c & -\sin b \\ \sin a \sin b \cos c - \cos a \sin c & \sin a \sin b \sin c + \cos a \cos c & \sin a \cos b \\ \cos a \sin b \cos c + \sin a \sin c & \cos a \sin b \sin c - \sin a \cos c & \cos a \cos b \end{bmatrix} \tag{3.4}$$

This parameterization ensures that the $\mathbf{R}_{all}$ will be a rotation matrix, so using it in the optimization process will always result in valid rotation matrices. However as we can see in equation (3.4), the parameterization depends on trigonometric functions that can be hard to solve efficiently and accurately. This claim was found to be true in practice with the implementation of the method.

The use of the above parameterization failed to find the global minimum most of the times. The optimization method gets stuck in local minima and in general converges very slowly. The optimization step was conducted in the beginning using forward differencing for the calculation of the Jacobian matrix. Analytical Jacobian was used also but we didn't observe significant improvements in terms of convergence time.

## 3.3 Results

It is obvious that this method works on all images at once. It basically requires a non-linear optimization step. The complexity of the problem can get quite big with growing number of images and point correspondences among the images. This makes the system slow, non robust and thus inappropriate for time-critical applications.

The implementation of the system revealed several flows in the method. The most important is that there is no efficient parametrization of the rotation matrices.

# Chapter 4

# An efficient method for three-dimensional reconstruction

## 4.1 Introduction

In this chapter we will describe an efficient way to reconstruct a scene based on the geometric properties of the *two view geometry*. We describe the steps to acquire a three-dimensional representation of a scene along with the theoretical background. First we present a way to estimate the intrinsic parameters of the camera, or in other words a method to calibrate the cameras. We then explore the constraints that are present when we have two views of a rigid scene, and we describe the so called *epipolar geometry.* We continue by exploiting the epipolar geometry to acquire the projection matrixes of the cameras and finally to solve the reconstruction problem with triangulation.

## 4.2 Camera calibration

Suppose that we have a camera by which we take pictures of a scene we wish to reconstruct in three-dimensions. It is important to know the way this particular camera formulates the images on the film or CCD sensor. This is equivalent to knowing the intrinsic parameters of the camera. A camera with known intrinsic parameters is referred as a *calibrated camera*, and the procedure of acquiring these parameters is called *camera calibration.* So we will first describe a method for calibrating a camera. We will present a method for estimating the intrinsic parameters of the camera as described in Section 2.4.2.

These parameters are different for every camera and depend on the manufacturing characteristics of the camera and the topological placement of the lens when the photographs where taken. These parameters can be available from the manufacturer, but that is not always the case. That means that we must device ways to obtain these parameters, through the process of calibration.

The calibration technique described here is based on work of Zhang [10, 11], and is based on viewing a planar object from different views.

### 4.2.1 The absolute conic

When dealing with camera calibration we first define the most basic parameters of the camera. Since little in known we have to work with an abstract object in the scene. The *absolute conic.* The absolute conic is a special conic that is positioned at the plane at infinity[1], which is invariant to transformations in three-dimensional space. Consider this like the effect the moon gives on a moving observer on earth. It seems like it is not moving in relation to the observer. Now expand this to include rotations and you will be able to grasp the concept of the absolute conic.

This has the advantage that the image of the absolute conic is independent of the position and orientation of the camera. The absolute conic is symbolized as $\omega_\infty$ and is related to the camera calibration matrix $\mathbf{K}$ in the following way:

$$\omega_\infty = \mathbf{K}^{-\mathsf{T}}\mathbf{K}^{-1} \tag{4.1}$$

[1]The *plane at infinity* is an augmentation of the Euclidian space and is the plane where all parallel lines meet. This plane consists of points with the last coordinate set to zero, in their homogeneous representation

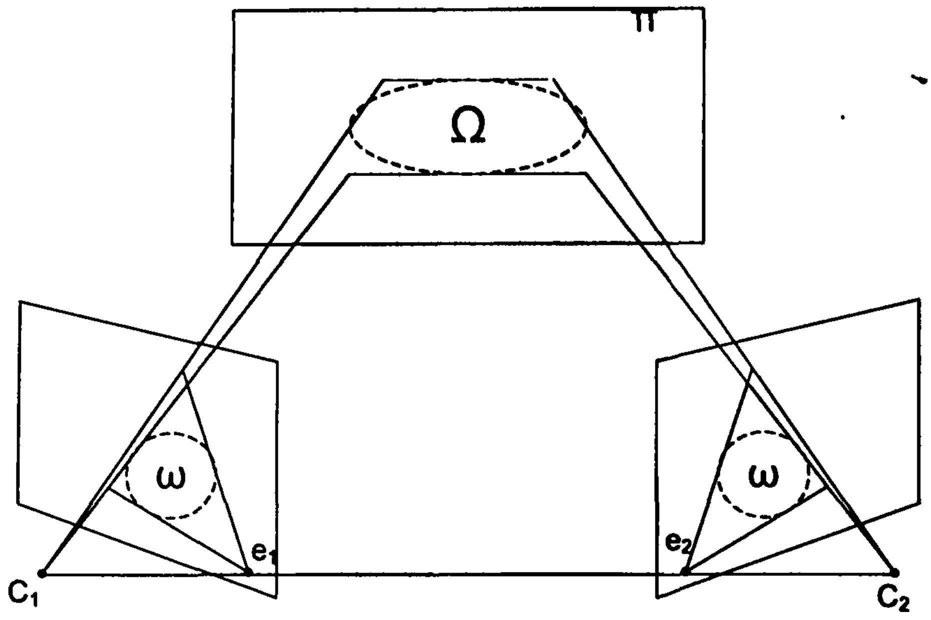


Figure 4.1: The images of the absolute conic

Therefore knowing the absolute conic is equivalent to knowing the cameras calibration matrix. This means that by estimating the absolute conic we can find the intrinsic parameters of the camera, and thus we have solved the camera calibration problem.

### 4.2.2 A planar object and its image

In this calibration technique we use the view of a calibration pattern consisting of a black and white checkerboard with squares of known length. In Figure 4.2 we can see the planar calibration object photographed from several positions. The calibration object consists of an A4 paper that was glued on a desk to ensure that it is planar. The printed black and white squares are 25mm×25mm. Since the object is planar we are able to establish a homography between it and its image.

We will assume with no loss of generality that the planar object lies on $Z = 0$ in the world coordinate system. Taking the projection equation (??) and representing the $i^{th}$ column vector of the rotation matrix by $\mathbf{r}_i$, we obtain the following equation:

$$s\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \qquad (4.2)$$

Now because $Z = 0$, the homogenous coordinates of point $\mathbf{X}$ are written as $\tilde{\mathbf{X}} = [X, Y, 1]^{\mathsf{T}}$. A planar object point $\mathbf{X}$ is related to its image point $\mathbf{x}$ by a $3\times3$ homography matrix $\mathbf{H}$:

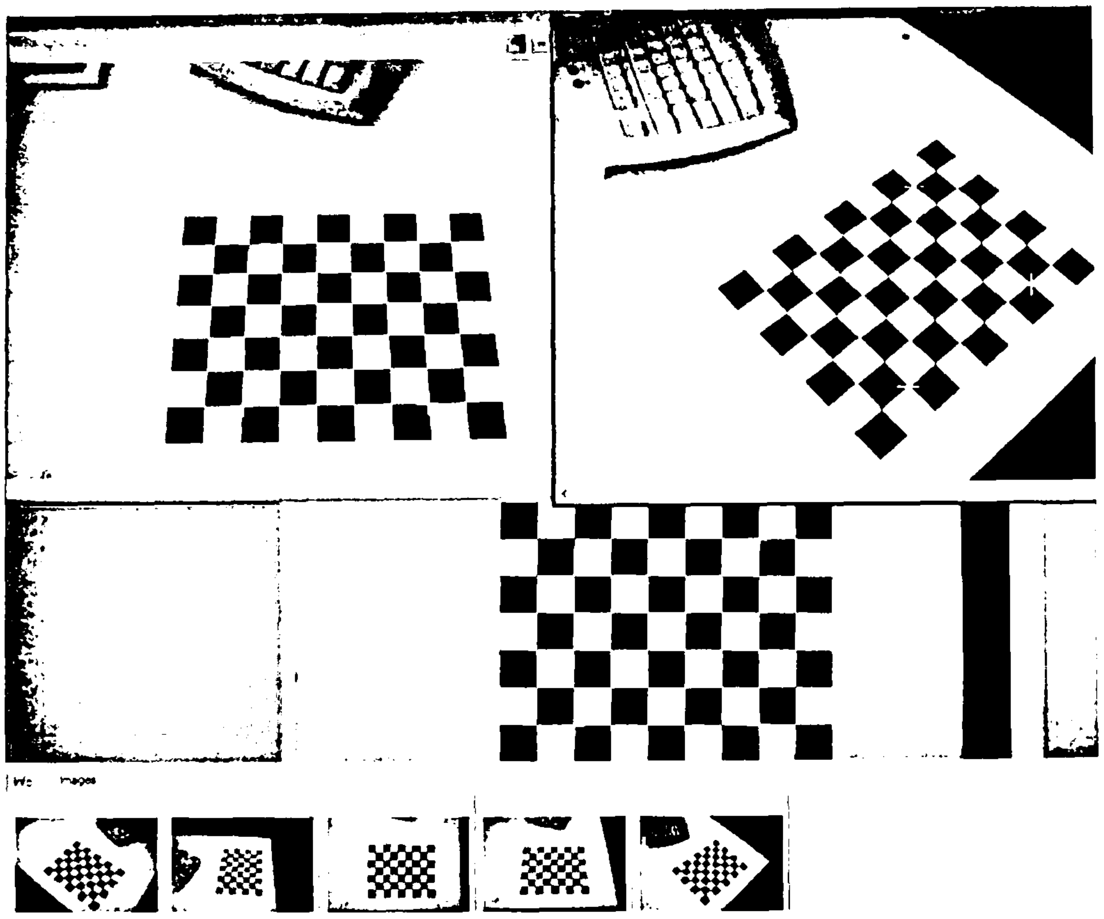


Figure 4.2: The planar calibration object photographed from various position

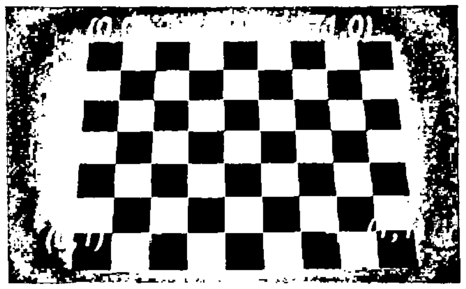


Figure 4.3: The selected world space coordinates for the four corners of the pl object

$$s\tilde{\mathbf{x}} = \mathbf{H}\tilde{\mathbf{X}} \tag{4.3}$$

with $\mathbf{H} = \mathbf{K}[\mathbf{r}_1 \;\; \mathbf{r}_2 \;\; \mathbf{t}]$

The homographies can be estimated by selecting the four corners on the calibration object, as shown in Figure 4.3. In that figure we can also see the world space coordinates $X_{1...4}$ used.

Zhang [10, 11] uses a maximum likelihood criterion to estimate the homography. The maximum likelihood criterion of $\mathbf{H}$ is obtained by minimizing the following:

$$\min_{\mathbf{H}} \sum_i ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 \tag{4.4}$$

where

$$\hat{\mathbf{x}}_i = \frac{1}{\mathbf{h}_3^\mathsf{T}\mathbf{X}_i} \begin{bmatrix} \mathbf{h}_1^\mathsf{T}\mathbf{X}_i \\ \mathbf{h}_2^\mathsf{T}\mathbf{X}_i \end{bmatrix} \tag{4.5}$$

with $\mathbf{h}_i$ the $i_{th}$ row of $\mathbf{H}$. This is a nonlinear minimization problem that can be solved using the *Levenberg-Marquardt* optimization algorithm.

By setting $\chi = [\mathbf{h}_1^\mathsf{T} \;\; \mathbf{h}_2^\mathsf{T}\mathbf{h}_3^\mathsf{T}]^\mathsf{T}$ the equation (4.3) can be written as follows:

$$\begin{bmatrix} \tilde{\mathbf{X}}^\mathsf{T} & \mathbf{0}^\mathsf{T} & -x\tilde{\mathbf{X}}^\mathsf{T} \\ \mathbf{0}^\mathsf{T} & \tilde{\mathbf{X}}^\mathsf{T} & -y\tilde{\mathbf{X}}^\mathsf{T} \end{bmatrix} \chi = 0 \tag{4.6}$$

If we have $n$ points, $n$ equations are obtained and can be written in a matrix form $\mathbf{L}\chi = 0$, where $\mathbf{L}$ is a $2n \times 9$ matrix. The solution is then derived as the eigenvector of $\mathbf{L}^\mathsf{T}\mathbf{L}$ associated with the smallest eigenvalue.

### 4.2.3 Estimating the camera calibration matrix

We will now try to estimate the camera calibration matrix. We start by writing $\mathbf{H} = [\mathbf{h}_1 \;\; \mathbf{h}_2 \;\; \mathbf{h}_3]$ in equation (4.3) :

$$[\mathbf{h}_1 \;\; \mathbf{h}_2 \;\; \mathbf{h}_3] = \lambda\mathbf{K}[\mathbf{r}_1 \;\; \mathbf{r}_2 \;\; \mathbf{t}] \tag{4.7}$$

where $\lambda$ is a scalar. It is true that the vectors $\mathbf{r}_1$ and $\mathbf{r}_2$ are orthonormal, since they are rows of the rotation matrix $\mathbf{R}$. Then the following two equations are obtained and give two constraints on the internal parameters of the camera:

$$\mathbf{h}_1^\mathsf{T}\mathbf{K}^{-\mathsf{T}}\mathbf{K}^{-1}\mathbf{h}_2 = 0 \tag{4.8}$$

$$\mathbf{h}_1^\mathsf{T}\mathbf{K}^{-\mathsf{T}}\mathbf{K}^{-1}\mathbf{h}_1 = \mathbf{h}_2^\mathsf{T}\mathbf{K}^{-\mathsf{T}}\mathbf{K}^{-1}\mathbf{h}_2 \tag{4.9}$$

It can be seen that $\mathbf{K}^{-\mathsf{T}}\mathbf{K}^{-1}$ represents the image of the absolute conic $\omega_\infty$.

We expand the equation (4.1) and a symmetric matrix is obtained:

$$\omega_\infty = \mathbf{K}^{-\top}\mathbf{K}^{-1} = \begin{bmatrix} \omega_1 & \omega_2 & \omega_4 \\ \omega_2 & \omega_3 & \omega_5 \\ \omega_4 & \omega_5 & \omega_6 \end{bmatrix} = \begin{bmatrix} \frac{1}{f_x^2} & -\frac{s}{f_x^2 f_y} & \frac{c_y s - c_x f_y}{f_x^2 f_y} \\ -\frac{s}{f_x^2 f_y} & \frac{s_2}{f_x^2 f_y^2} + \frac{1}{f_y^2} & -\frac{s(c_y s - c_x f_y)}{f_x^2 f_y^2} - \frac{c_y}{f_y^2} \\ \frac{c_y s - c_x f_y}{f_x^2 f_y} & -\frac{s(c_y s - c_x f_y)}{f_x^2 f_y^2} - \frac{c_y}{f_y^2} & \frac{(c_y s - c_x f_y)^2}{f_x^2 f_y^2} + \frac{c_y}{f_y^2} + 1 \end{bmatrix} \tag{4.10}$$

Defining $\omega_u = [\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6]^\top$ and $\mathbf{h}_i = [h_{i1}, h_{i2}, h_{i3}]^\top$ the $i^{th}$ column of $\mathbf{H}$, the following equation is derived:

$$\mathbf{h}_i^\top \omega_\infty \mathbf{h}_j = \mathbf{u}_{ij}^\top \omega_u \tag{4.11}$$

where

$$\mathbf{u}_{ij} = \begin{bmatrix} h_{i1}h_{j1} & h_{i1}h_{j2} + h_{i2}h_{j1} & h_{i2}h_{j2} & h_{i3}h_{j1} + h_{i1}h_{j3} & h_{i3}h_{j2} + h_{i2}h_{j3} & h_{i3}h_{j3} \end{bmatrix}^\top \tag{4.12}$$

It then possible to rewrite the two constraint equations (4.8) and (4.9) as two homogeneous equations in $\omega_u$:

$$\begin{bmatrix} \mathbf{u}_{12}^\top \\ (\mathbf{u}_{11} - \mathbf{u}_{22})^\top \end{bmatrix} \omega_u = 0 \tag{4.13}$$

For $n$ images on $n$ homographies, the above vector equation is stacked $n$ times and the following is obtained:

$$\mathbf{V}\omega_u = 0 \tag{4.14}$$

with $\mathbf{V}$ being a $2n \times 6$ matrix. The general solution is then derived as the eigenvector of $\mathbf{V}^\top\mathbf{V}$ associated with the smallest eigenvalue. If only two images are present, it is possible to assume that the skew $s$ is zero (which is very common). This will be added as an additional row in $\mathbf{V}$. We can assume further that the principal point is at the image's center and solve with only one image.

The absolute conic $\omega_\infty$ is defined up to a scale, and it is possible to extract the intrinsic parameters of the camera, once vector $\omega_u$ is known:

$$c_y = \frac{\omega_2\omega_4 - \omega_1\omega_5}{\omega_1\omega_3 - \omega_2^2} \quad (4.15)$$

$$\lambda = \omega_6 - \frac{\omega_4^2 + c_y(\omega_2\omega_4 - \omega_1\omega_5)}{\omega_1} \quad (4.16)$$

$$f_x = \sqrt{\frac{\lambda}{\omega_1}} \quad (4.17)$$

$$f_y = \sqrt{\frac{\lambda\omega_1}{\omega_1\omega_3 - \omega_2^2}} \quad (4.18)$$

$$s = -\frac{\omega_2 f_x^2 f_y}{\lambda} \quad (4.19)$$

$$c_x = \frac{s c_x}{\lambda} - \frac{\omega_4 f_x^2}{\lambda} \quad (4.20)$$

Once the calibration matrix is calculated the external parameters for each image can be calculated also from the equation (4.3).

$$\mathbf{r1} = \lambda\mathbf{K}^{-1}\mathbf{h}_1 \quad (4.21)$$

$$\mathbf{r2} = \lambda\mathbf{K}^{-1}\mathbf{h}_2 \quad (4.22)$$

$$\mathbf{r3} = \mathbf{r1} \times \mathbf{r2} \quad (4.23)$$

$$\mathbf{t} = \lambda\mathbf{K}^{-1}\mathbf{h}_3 \quad (4.24)$$

where the scalar $\lambda = \frac{1}{||\mathbf{K}^{-1}\mathbf{h}_1||} = \frac{1}{||\mathbf{K}^{-1}\mathbf{h}_2||}$.

The solution we obtain from the above procedure is used as an initial guess to the a nonlinear optimization problem. We minimize:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} ||\mathbf{x}_{ij} - \hat{\mathbf{x}}(\mathbf{K}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)||^2 \quad (4.25)$$

Where $\hat{\mathbf{x}}(\mathbf{K}, \mathbf{R}_i, \mathbf{t}_i, \mathbf{X}_j)$ is the projection of point $\mathbf{X}_j$ in image $i$. This optimization is solved with the *Levenberg-Marquardt* non linear least squares optimization algorithm.

## 4.3 Epipolar geometry

One important question that someone can ask is whether there is a constraint between a pair of perspective images of a scene. It turns out that there is such a constraint, the *epipolar constraint*. This constraint states that for each point in one of the images, the corresponding point in the other image must lie on a straight line.

This constraint is expressed mathematically by a $3 \times 3$ singular matrix, known as the *fundamental matrix* and denoted by **F**. If a three-dimensional point **X** is projected at point **x** in the first image and at point **x**′ at the second one, then the image points satisfy the relation:

$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0 \tag{4.26}$$

The fundamental matrix in essence describes the *epipolar geometry* between two views of the same scene. The epipolar geometry is the geometry of the intersection of the image retinal planes with the pencil of planes having as axis the line joining the camera centers, as seen in Figure 4.5.

Lets consider two photographs of the same scene taken from two distinct points in space. If the cameras are *calibrated* and have a matrix **K** equal to **I**, the homogeneous coordinates **x** and the spatial coordinates **X** of a point $p$, with respect to the camera frame:

$$\lambda\mathbf{x} = \Pi_0\mathbf{X} \tag{4.27}$$

That means that the image **x** differs from the actual three-dimensional coordinates of the point by a depth $\lambda \in \mathbb{R}_+$. Now without loss of generality we can assume that the world frame is aligned with the first camera, while the other camera is positioned and oriented according to an Euclidian transformation $g = (\mathbf{R}, \mathbf{t})$. Suppose now that the three-dimensional coordinates of a point $p$ relative to the two cameras are $\mathbf{X}_1, \mathbf{X}_2 \in \mathbb{R}^3$, they are related by the following transformation:

$$\mathbf{X}_2 = \mathbf{R}\mathbf{X}_1 + \mathbf{t} \tag{4.28}$$

Now let the $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^3$ be the homogeneous coordinates of the projection of the same point $p$ in the two photographs. Since $\mathbf{X}_i = \lambda_i\mathbf{x}_i, i = 1, 2$, the above equation can be written as:

$$\lambda_2\mathbf{x}_2 = \mathbf{R}\lambda_1\mathbf{x}_1 + \mathbf{t} \tag{4.29}$$

In order to eliminate the depths $\lambda_i$ we multiply both sides with $[t]_\times$:

$$\lambda_2[\mathbf{t}]_\times\mathbf{x}_2 = [\mathbf{t}]_\times\mathbf{R}\lambda_1\mathbf{x}_1 \tag{4.30}$$

Since the vector $[\mathbf{t}]_\times\mathbf{x}_2 = \mathbf{t} \times \mathbf{x}_2$ is perpendicular to the vector $\mathbf{x}_2$, their inner product is zero. Multiplying the previous equation with $\mathbf{x}_2^{\mathrm{T}}$ yields that the quantity $\mathbf{x}_2^{\mathrm{T}}[\mathbf{t}]_\times\mathbf{R}\lambda_1\mathbf{x}_1$ is zero. Since $\lambda_1 > 0$, we have proven the following:

**Theorem 4.3.1.** *Consider two images* $\mathbf{x}_1, \mathbf{x}_2$ *of the same point* $p$ *from two camera positions with relative pose* $(\mathbf{R}, \mathbf{t})$*, where* **R** *the relative orientation and* **t** *the relative position. Then* $\mathbf{x}_1, \mathbf{x}_2$ *satisfy:*

$$\mathbf{x}_2^{\mathrm{T}}[t]_\times\mathbf{R}\mathbf{x}_1 = 0 \tag{4.31}$$

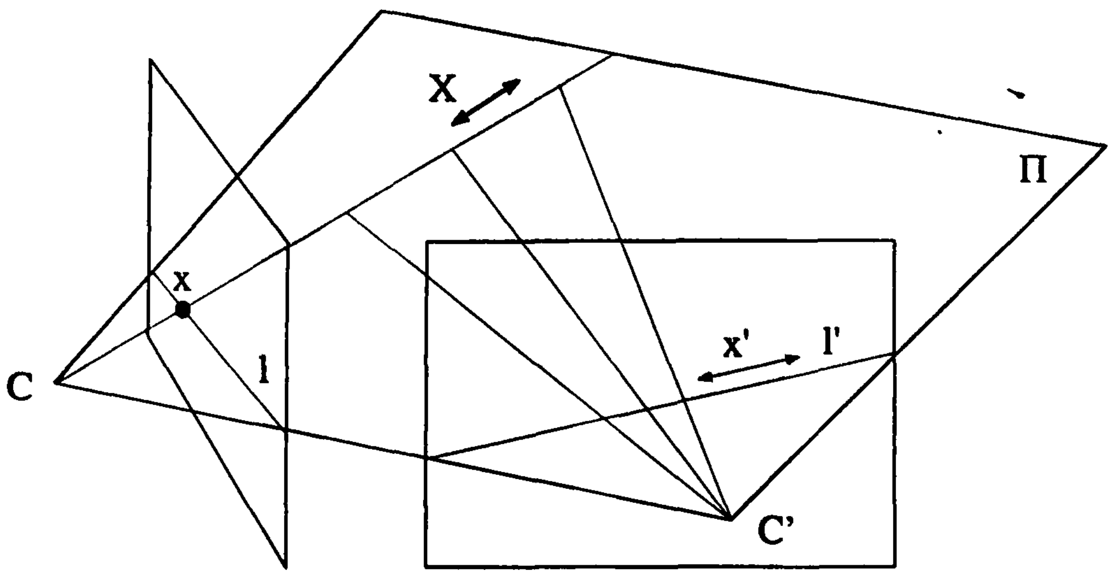


Figure 4.4: Epipolar correspondence

The matrix $\mathbf{E} = [t]_{\times}\mathbf{R}$ (as we will see later on) is called the *Essential matrix* and is a specialization of the *fundamental matrix* for the case of calibrated cameras.

To understand the epipolar geometry and the epipolar constraint, lets discuss about the point projection properties. Given the projection point **x** on one image, we are not able to know the exact location of **X** but we know that is bound to be on the line of sight of **x**. Check the Figure 4.4. The line of sight of **x** is the line joining the center of projection C of the camera and **x**. This line can be projected on another image and the corresponding image point $\mathbf{x}'$ is bound to be on the projected line $\mathbf{l}'$. In fact all the points on plane Π defined by the two projection centers and the point **X** have their image on $\mathbf{l}'$. In the same fashion all these points are projected on the first image on a line **l**. There two lines are then in *epipolar correspondence*.

Suppose now that we have more three-dimensional points in the scene. With every three-dimensional point and the centers of projections, we get a different plane. As we can see in Figure 4.5 each such plane results in a pair of corresponding epipolar lines. All these lines pass through two specific points e and $\mathbf{e}'$. These special points are called the *epipoles* and they are the projections of the center of projection of one image to the other.

These concepts were introduced by Faugeras [2] and Hartley [3]. Since then, there has been a lot of research on the properties of the fundamental matrix and the methods for estimating it from two uncalibrated images [4].

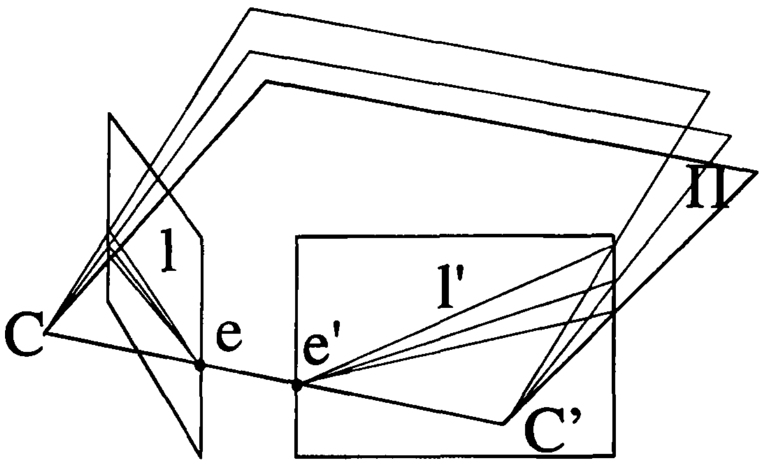


Figure 4.5: Epipolar geometry

## 4.4 Estimating the fundamental matrix

The Equation (4.26) can be used to estimate the fundamental matrix for two uncalibrated images. Every pair of corresponding image points gives one constraint on the fundamental matrix **F**. Since **F** is a $3 \times 3$ matrix, we have nine unknowns. But since we require that the matrix is of rank 2, we need to determine it only up to scale so we have eight unknowns. Therefore 8 pairs of corresponding image points are sufficient to compute **F**.

### 4.4.1 The eight-point algorithm

The most straightforward method for estimating the fundamental matrix is using Equation (4.26) with eight corresponding image points and solving a linear system. Equation (4.26) can be written as:

$$[xx' \;\; yx' \;\; x' \;\; xy' \;\; yy' \;\; y' \;\; x \;\; y \;\; 1]\,\mathbf{f} = 0 \tag{4.32}$$

where $\mathbf{x} = [x\ y\ 1]^{\mathsf{T}}$, $\mathbf{x}' = [x'\ y'\ 1]^{\mathsf{T}}$ and $\mathbf{f} = [F_{11}\ F_{12}\ F_{13}\ F_{21}\ F_{22}\ F_{23}\ F_{31}\ F_{32}\ F_{33}]^{\mathsf{T}}$ with $F_{ij}$ being the elements of the fundamental matrix **F**. By stacking eight of these equations in a matrix **A**, we obtain the following equation:

$$\mathbf{Af} = 0 \tag{4.33}$$

This system of equations can be solved by *Singular Value Decomposition(SVD)*. The SVD of **A** results the decomposition $\mathbf{USV}^{\mathsf{T}}$ with **U** and **V** orthonormal matrices and **S** a diagonal matrix containing the singular values. The singular values $\sigma_i$ are positive and decreasing in order. In our case $\sigma_9$ is guaranteed to be zero and thus the last column of **V** is the solution. Of course, this holds as long as the eight equations are linearly independent[2].

Here we must note that in the presence of noise on the points coordinates, the estimated matrix will not satisfy the rank 2 constraint. This constraint is very important and most applications of the fundamental matrix rely on the fact that it is rank 2. The most convenient way is to enforce the constraint after the initial solution is obtained. To do so we replace the matrix $\mathbf{F}$ with the matrix $\mathbf{F}'$ that minimizes the Frobenius norm $\|\mathbf{F}-\mathbf{F}'\|$ subject to the condition $\det \mathbf{F}' = 0$. To do this, let $\mathbf{F} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$ be the SVD of $\mathbf{F}$, with $\mathbf{D} = diag(r, s, t)$. We then let $\mathbf{F}' = \mathbf{U}diag(r, s, 0)\mathbf{V}^{\mathsf{T}}$. This method was suggested by Tsai and Huang [5] and has been proven to minimize the Frobenius norm of $\mathbf{F} - \mathbf{F}'$ as required.

## 4.4.2 Improving the eight-point algorithm

We went on and improved this fundamental matrix estimation algorithm by applying a normalization on the input point coordinates [9]. What we do is we transform the image points before we feed them in the eight-point algorithm. Suppose that coordinates $\mathbf{x}$ in one image are replaced by $\hat{\mathbf{x}} = \mathbf{T}\mathbf{x}$, and coordinates $\mathbf{x}'$ in the second image are replaced by $\hat{\mathbf{x}'} = \mathbf{T}'\mathbf{x}'$. Substituting that to equation (4.26) we get :

$$\hat{\mathbf{x}}'^{\mathsf{T}}\mathbf{T}'^{-\mathsf{T}}\mathbf{F}\mathbf{T}^{-1}\hat{\mathbf{x}} = 0 \tag{4.34}$$

where $\mathbf{T}'^{-\mathsf{T}}$ is the inverse transpose of $\mathbf{T}'$. This relation implies that $\mathbf{T}'^{-\mathsf{T}}\mathbf{F}\mathbf{T}^{-1}$ is the fundamental matrix corresponding to the transformed points. So what we do is :

1. Transform the image coordinates according to transformations $\hat{\mathbf{x}} = \mathbf{T}\mathbf{x}$ and $\hat{\mathbf{x}}' = \mathbf{T}'\mathbf{x}'$
2. Estimate the fundamental matrix $\hat{\mathbf{F}}$ corresponding to the transformed points
3. Set $\mathbf{F} = \mathbf{T}'^{\mathsf{T}}\hat{\mathbf{F}}\mathbf{T}$

The fundamental matrix estimated this way will correspond to the original untransformed points no matter what that transformation was. Hartley [9] showed that using a specific transformation can result in the eight-point algorithm to perform almost as good as the algorithms that use non-linear criterions.

This transformation is done in two steps. The first translates the coordinates in each image so that the centroid of the point set is brought to the origin. The coordinates are then scaled, the suggested scaling is such that the average point is $[1, 1, 1]^{\mathsf{T}}$. Such point will lie at a distance $\sqrt{2}$ from the origin. So the transformation is as follows :

1. The points are translated so that their centroid is at the origin.

[2]All other singular values must be non zero

2. The points are scaled isotropically for both coordinates so that the averag tance from the origin is equal to $\sqrt{2}$

The transformation is applied on the two images independently.

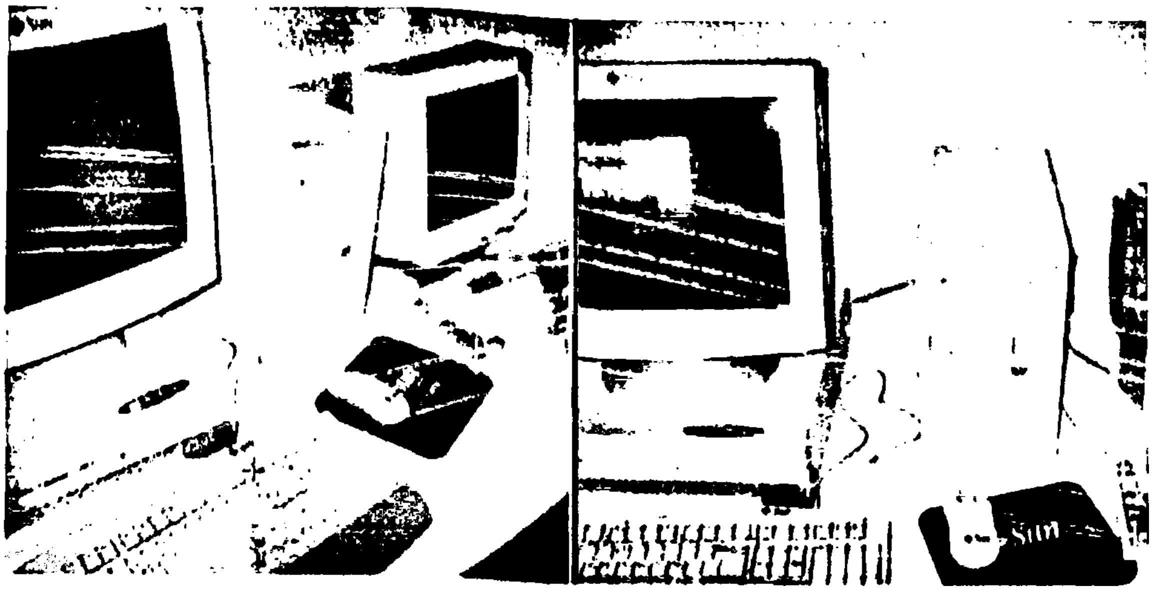


Figure 4.6: The epipolar geometry estimated by the eight-point algorithm wi normalization

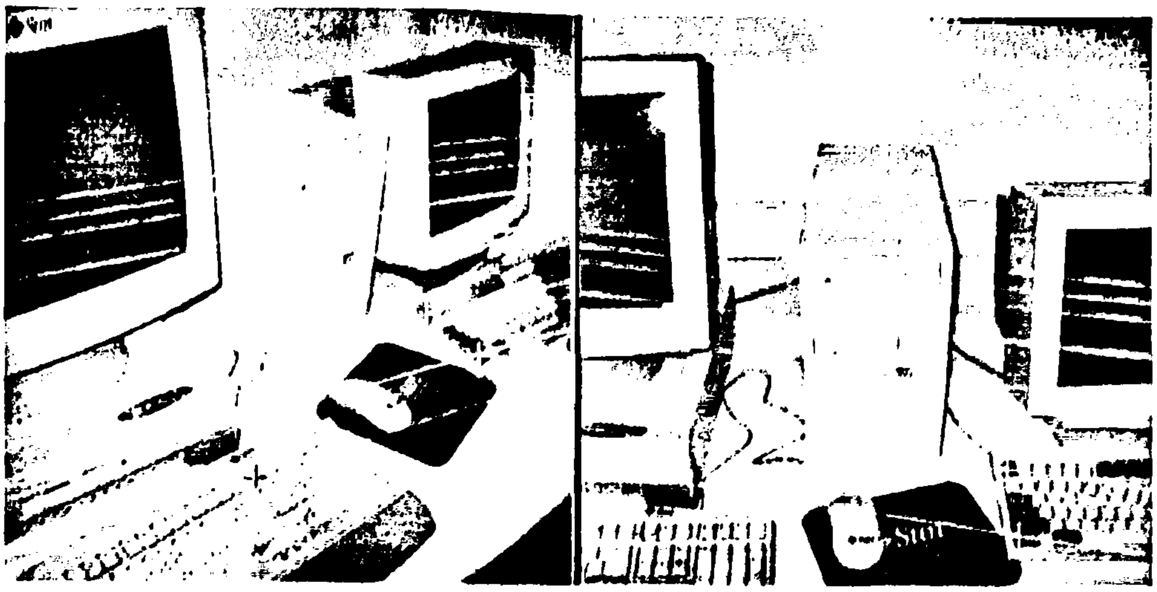


Figure 4.7: The epipolar geometry estimated by the eight-point algorithm wit malization

In Figure 4.6 and Figure 4.7 we can see the results of the point coordinates nc ization. The epipolar geometry in figure 4.7 is good and is almost identical to the non-linear criterion algorithm gives. On the other hand the epipolar geome 4.6 given by the eight-point algorithm without normalization of the point coord is distorted.

### 4.4.3 Distance minimization algorithm

In general, it is possible to obtain more than eight image point correspondences from two images. It that case, we should use the extra points in order to minimize the effect of noise on the result. The original eight-point algorithm can be extended to use more points, in a linear least squares fashion. In that case the matrix **A** of equation (4.33) will contain rows for all additional point matches. The solution is obtained in a similar way. The last singular value is used again, only that now it is not zero.

Even though the above method is fast and simple to implement, it is very sensitive to noise, even when there are more than eight point correspondences. We will discuss this matter in the implementation chapter of this thesis but one of the main problems with this method is that we minimize an algebraic error which does not have a *"physical"* meaning. It would be better if we where minimizing a more geometrically meaningful criterion.

The obvious error that we should be minimizing is the distance of points from the corresponding epipolar lines. So the first idea is to use the following non-linear criterion:

$$\sum_i d^2\left(\mathbf{x}'_i, \mathbf{F}\mathbf{x}_i\right).$$

with $d(\mathbf{x}, \mathbf{l})$ being the euclidian distance of point $\mathbf{x}$ from line $\mathbf{l} = [l_1, l_2, l_3]^\top$:

$$d(\mathbf{x}, \mathbf{l}) = \frac{|\mathbf{x}^\top \mathbf{l}'|}{\sqrt{(l_1)^2 + (l_2)^2}} \tag{4.35}$$

The problem with the above criterion is that, unlike the linear criterion of the eight-point algorithm, the two images does not play a symmetric role. This criterion determines only the epipolar lines in the second image.

To have a consistent fundamental matrix it is necessary and sufficient that by exchanging the two images, the fundamental matrix is changed to its transpose. So we must include that in the minimization criterion, hence we minimize this quantity:

$$\sum_i \left(d^2\left(\mathbf{x}'_i, \mathbf{F}\mathbf{x}_i\right) + d^2\left(\mathbf{x}_i, \mathbf{F}^\top\mathbf{x}'_i\right)\right)$$

### 4.4.4 Singularity constraint

The problem with the previous method is that we do not take into account the fact that **F** is of rank 2. We could use minimizations under the constraint $\det(\mathbf{F}) = 0$. This constraint is a cubic polynomial in the coefficients of **F** and the numerical implementations are not efficient enough to be usable.

It turns out that we can enforce the singularity constraint by using a certain parameterization. The idea is to express the matrix **F** as:

$$\mathbf{F} = \begin{bmatrix} x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 \\ x_7x_1 + x_8x_4 & x_7x_2 + x_8x_5 & x_7x_3 + x_8x_6 \end{bmatrix} \quad (4.36)$$

What we basically do is write the third row of the matrix as a linear combination of the two first rows. This ensures that the matrix will be singular.

The previous parameterization takes into account only the fact that **F** is singular. It would be appropriate to parameterize it by values that are more significant to us. As Quang-Tuan Luong [6] we can use this parameterization:

$$\mathbf{F} = \begin{bmatrix} a & b & -ax - by \\ c & d & -cx - dy \\ -ax' - cy' & -bx' - dy' & (ax + by)x' + (cx + dy)y' \end{bmatrix} \quad (4.37)$$

Where, $x$ and $y$ are the coordinates of the first epipole, $x'$ and $y'$ are the coordinates of the second epipole, and $a$,$b$,$c$ and $d$ parameterize the epipolar transformation mapping an epipolar line in the first image to its corresponding epipolar line in the second image.

With this parameterization and the constraint of equation (4.4.3) we can use a non-linear optimization method to obtain the fundamental matrix for two arbitrary views, based on point correspondences.

## 4.5 Normalized coordinates

Consider a camera projection matrix as described in equation (2.15):

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}]$$

Suppose that the calibration matrix **K** is known. Now if $\mathbf{x} = \mathbf{PX}$ is a point on the image, we can apply $\mathbf{K}^{-1}$ to the point $\mathbf{x}$ to obtain:

$$\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{x} \quad (4.38)$$

Then $\hat{\mathbf{x}}$ is image point expressed in *normalized coordinates.* It may be thought of as the projection of a point $\mathbf{x}$ with respect to a camera having as a calibration matrix the identity matrix **I**.

The camera projection matrix:

$$\mathbf{K}^{-1}\mathbf{P} = [\mathbf{R}|\mathbf{t}]$$

is called *normalized camera matrix* as the effect of the known calibration matrix has been removed.

## 4.6 The essential matrix

The essential matrix is a specialization of the fundamental matrix. It is basically the fundamental matrix in the case of normalized coordinates. The definition of the essential matrix is:

$$\hat{\mathbf{x}}'^{\top}\mathbf{E}\hat{\mathbf{x}} = 0 \tag{4.39}$$

in terms of the normalized image coordinates for the corresponding points $\mathbf{x}$, $\mathbf{x}'$. Substituting $\hat{\mathbf{x}}$ and $\hat{\mathbf{x}}'$ with equation (4.38) gives:

$$\mathbf{x}'^{\top}\mathbf{K}'^{-\top}\mathbf{E}\mathbf{K}^{-1}\mathbf{x} = 0$$

Comparing this with the relation (4.26) for the fundamental matrix, it follows that the relationship between the fundamental matrix and essential matrix is:

$$\mathbf{E} = \mathbf{K}'^{\top}\mathbf{F}\mathbf{K} \tag{4.40}$$

## 4.7 Image plane homographies

Another important concept in projective geometry is the *plane homography.* A plane homography is a nonsingular $3 \times 3$ matrix which relates two uncalibrated retinal images of a three-dimensional plane.

If $\mathbf{x}$ is the projection of a point on a plane on one image and $\mathbf{x}'$ is the projection of the same point on the second image, then the two projections are related by the linear projection transformation:

$$\mathbf{x}' \sim \mathbf{H}\mathbf{x} \tag{4.41}$$

Shashua shows that the fundamental matrix and plane homographies are tightly coupled [7]. To be more exact, the entire group of all possible homography matrixes between two images lies in a subspace of dimension 4. It is spanned by four homography matrixes. These four homography matrixes are such that their respective planes to not all coincide with a single point. It is shown [8] that given the fundamental matrix $\mathbf{F}$ of an image pair, a suitable basis of four homography matrixes $\mathbf{H}_1, ..., \mathbf{H}_4$, referred as "primitive homographies", is defined as follows:

$$\mathbf{H}_i = [\epsilon_i]_\times \mathbf{F}, \quad i = 1, 2, 3$$

and

$$\mathbf{H}_4 = \mathbf{e}'\delta^{\top} \tag{4.42}$$

where $\epsilon_i$ are the identity vectors :

$$\epsilon_1 = [1,0,0]^\top, \epsilon_2 = [0,1,0]^\top, \epsilon_3 = [0,0,1]^\top$$

$[.]_\times$ designates the skew symmetric matrix representing the vector cross product, so that for a vector $\mathbf{x}$, $[\mathbf{x}]_\times \mathbf{b} = \mathbf{x} \times \mathbf{b}, \quad \forall\ \mathbf{b}$. More specifically:

$$[\mathbf{x}]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} \tag{4.43}$$

$\delta$ is a vector such that $\delta^\top \mathbf{e} \neq 0$. This can be satisfied by defining vector $\delta$ so that each of its elements has an absolute value of one and a sign identical to that of the corresponding element of $\mathbf{e}$.

The first three homography matrixes are of rank 2 and span the subgroup of homographies matrices whose associated plane contains the center of projection $\mathbf{C}'$ of the second camera. The fourth homography matrix has an associated plane that goes through the center of projection $\mathbf{C}$ of the first camera and not coincident with with $\mathbf{C}'$, thus having rank 1. The four primitive homographies allows any other homography $\mathbf{H}$ to be expressed as a linear combination of them:

$$\mathbf{H} = \sum_{i=1}^{4} \lambda_i \mathbf{H}_i \tag{4.44}$$

## 4.8 Extracting structure and motion

Until now we explored how different views of the same scene relate to each other. In this section we will try to use these relations to estimate the structure of the scene and the motion of the camera.

Contrary to the method described in Section 3.2 where we derived both scene structure and camera motion simultaneously in a big optimization problem, here we will see how we can obtain the camera motion alone at first, and then use that information along with the point correspondences to estimate the structure of the scene. So this method extracts motion and structure in two separate steps.

### 4.8.1 Camera motion

In order to estimate the motion of the cameras we will have to use corresponding image points from the images. So we will suppose we have a number of corresponding image point pairs. Given these image points we will be able to construct the epipolar geometry constraint. This means that we will estimate the fundamental matrix $\mathbf{F}$ for the given pair of images.

In the following procedure we can use the fundamental matrix to estimate the motion of the cameras, but this will give results only up to a projective ambiguity. On the other hand if we use the essential matrix the camera matrices may be retrieved from up to scale and a four-fold ambiguity. That is that there are four possible solutions, each of which at an undetermined scale.

### 4.8.2 Using the essential matrix to extract the projection matrixes

With no loss of generality we will assume that the first camera's projection matrix is $\mathbf{P} = [\mathbf{I}|\mathbf{0}]$. This mean that we position the first camera to be aligned with the world frame. In order to compute the second camera's projection matrix $\mathbf{P}'$, it is necessary to factor the essential matrix $\mathbf{E}$ into a product $\mathbf{SR}$ of a skew symmetric matrix and a rotation matrix.

We will use the matrices:

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \tag{4.45}$$

note that $\mathbf{W}$ is orthogonal and $\mathbf{Z}$ is skew-symmetric. Suppose now that the SVD of E is $\mathbf{U}diag(1,1,0)\mathbf{V}^{\mathsf{T}}$. There are two possible $\mathbf{SR}$ factorizations of the essential matrix, as follows:

$$\mathbf{S} = \mathbf{UZU}^{\mathsf{T}} \quad \mathbf{R} = \mathbf{UWV}^{\mathsf{T}} \ \ or \ \ \mathbf{UW}^{\mathsf{T}}\mathbf{V}^{\mathsf{T}} \tag{4.46}$$

The given factorization is true, and it is verifiable by inspection. The above factorization determines the translation part $\mathbf{t}$ of the camera's projection matrix $\mathbf{P}'$, up to scale, from $\mathbf{S} = [\mathbf{t}]_{\times}$. However, the Frobenius norm of $\mathbf{S} = \mathbf{UZU}^{\mathsf{T}}$ is $\sqrt{2}$, which means that if $\mathbf{S} = [\mathbf{t}]_{\times}$ including scale then $||\mathbf{t}|| = 1$, which is a convenient normalization for the baseline of the two projection matrixes. Since $\mathbf{St} = 0$, it means that $\mathbf{t} = \mathbf{U}[0,0,1]^{\mathsf{T}} = \mathbf{u}_3$, the last column of $\mathbf{U}$.

However the sign of E and consequently $\mathbf{t}$, cannot be determined. Therefore given an essential matrix, there are four possible choices for the second camera projection matrix. Two for the possible choices of $\mathbf{R}$ and another two for the possible signs of $\mathbf{t}$.

So the available choices for the second projection matrix are:

$$\begin{aligned} \mathbf{P}' &= [\mathbf{UWV}^{\mathsf{T}}| + \mathbf{u}_3] \\ \mathbf{P}' &= [\mathbf{UWV}^{\mathsf{T}}| - \mathbf{u}_3] \\ \mathbf{P}' &= [\mathbf{UW}^{\mathsf{T}}\mathbf{V}^{\mathsf{T}}| + \mathbf{u}_3] \end{aligned}$$

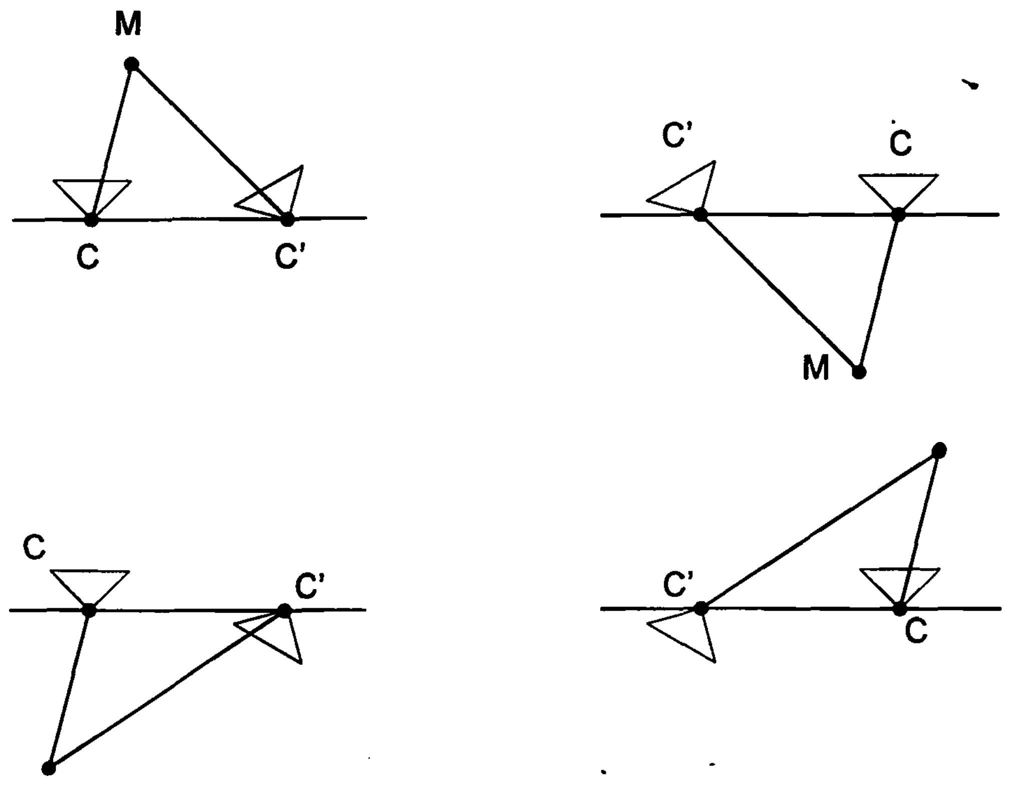


Figure 4.8: The four possible solutions for the projection matrixes

$$\mathbf{P}' = [\mathbf{U}\mathbf{W}^\top\mathbf{V}^\top| - \mathbf{u}_3] \tag{4.47}$$

### 4.8.3 The geometrical interpretation

It is obvious that the difference between the first two solutions is simply that the direction of the translation vector is the opposite. For the relation between the first and third projection matrixes we have to pay some more attention. It is verifiable that:

$$[\mathbf{U}\mathbf{W}^\top\mathbf{V}^\top|\mathbf{u}_3] = [\mathbf{U}\mathbf{W}\mathbf{V}^\top|\mathbf{u}_3]\begin{bmatrix}\mathbf{U}\mathbf{W}^\top\mathbf{W}^\top\mathbf{V}^\top & \\ & 1\end{bmatrix} \tag{4.48}$$

and $\mathbf{U}\mathbf{W}^\top\mathbf{W}^\top\mathbf{V}^\top = \mathbf{V}diag(-1,-1,1)\mathbf{V}^\top$ is a rotation of 180° about the line joining the two centers of projection of the cameras.

In Figure 4.8 we can see the four solutions that we acquire from the essential matrix. In this figure the left and right sides present the translation reversal, and top and bottom subfigures present the 180° rotation around the line joining the two centers of projection. To determine which one of the four solutions is the correct one we will have to reconstruct a three-dimensional point from the point correspondences. This is further explained in the next section, but once we have reconstructed the three-dimensional point as shown in Figure 4.8 we can determine the correct solution by checking in which case the point is in front of both cameras. In the figure example

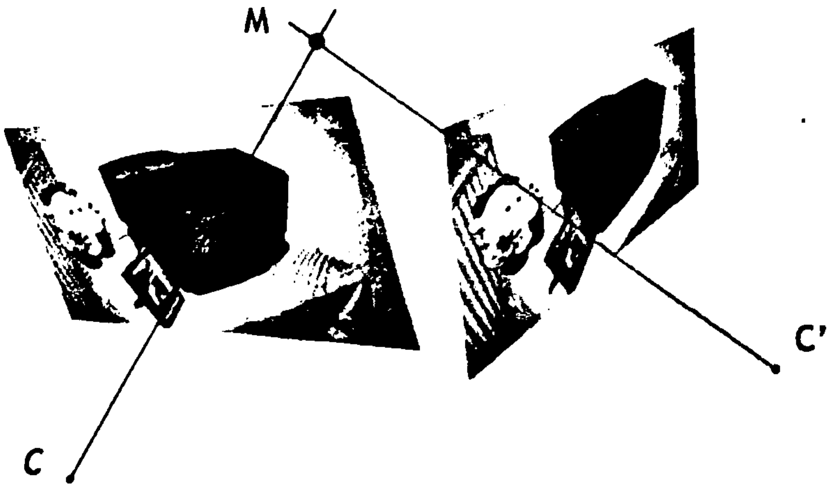


Figure 4.9: Acquiring three-dimensional point with triangulation

note that the reconstructed point $M$ is in front of both cameras only at the top-left subfigure.

Thus, testing with a single point to determine if it is in front of both cameras is sufficient to decide between the four available solutions for the second camera matrix.

### 4.8.4 Structure

Once the two projection matrixes have been determined we can use the available point matches to reconstruct actual three-dimensional points of the scene.

For each point on an image there is a corresponding line of sight that can be placed in space. This line is the line passing from the center of projection of the camera and the point on the retinal plane. In the case of two corresponding points on two different views, the lines of sight intersect in a three-dimensional point, that is the original scene point. This illustrated in Figure 4.9.

In practice, however, these lines will not perfectly intersect. This is because of faulty positioned points and the presence of noise.

So what we have are the two projection matrixes of the two views, along with the coordinates the point is represented on the retinal planes. So suppose that the three-dimensional point in space is $\mathbf{X} = [X, Y, Z, W]^\mathsf{T}$ and its projections on the images are $\mathbf{x} = [x, y, 1]^\mathsf{T}$ and $\mathbf{x}' = [x', y', 1]^\mathsf{T}$. By making use of the pinhole camera model equation (??) the following two equations can be defined:

$$s_1\mathbf{x} = \mathbf{P}_1\mathbf{X} \tag{4.49}$$

$$s_2\mathbf{x}' = \mathbf{P}_2\mathbf{X} \tag{4.50}$$

where $s_1$ and $s_2$ are two arbitrary scalars. Suppose now that $\mathbf{p}_{1i}^\mathsf{T}$ and $\mathbf{p}_{2i}^\mathsf{T}$ are the

$i^{th}$ row of $\mathbf{P}_1$ and $\mathbf{P}_2$ respectively. The scalars $s_1$ and $s_2$ are eliminated by setting the following: $s_1 = \mathbf{p}_{13}^\top \mathbf{X}$ and $s_2 = \mathbf{p}_{23}^\top \mathbf{X}$. Then the above equations can be written in the form:

$$\mathbf{A}\mathbf{X} = 0 \tag{4.51}$$

with $\mathbf{A}$ being a $4 \times 4$ matrix:

$$\mathbf{A} = \begin{bmatrix} \mathbf{p}_{11}^\top - x\mathbf{p}_{13}^\top \\ \mathbf{p}_{12}^\top - y\mathbf{p}_{13}^\top \\ \mathbf{p}_{21}^\top - x'\mathbf{p}_{23}^\top \\ \mathbf{p}_{22}^\top - y'\mathbf{p}_{23}^\top \end{bmatrix} \tag{4.52}$$

Then the solution is the eigenvector of the matrix $\mathbf{A}^\top\mathbf{A}$ associated with the smallest eigenvalue.

To further improve the result obtained from the linear system solution above, we can use a nonlinear optimization step. We minimize the error measured in the retinal plane between the observation and the projection of the reconstructed point [16]:

$$\left(x - \frac{\mathbf{p}_{11}^\top \mathbf{X}}{\mathbf{p}_{13}\mathbf{X}}\right)^2 + \left(y - \frac{\mathbf{p}_{12}^\top \mathbf{X}}{\mathbf{p}_{13}\mathbf{X}}\right)^2 + \left(x' - \frac{\mathbf{p}_{21}^\top \mathbf{X}}{\mathbf{p}_{23}\mathbf{X}}\right)^2 + \left(y' - \frac{\mathbf{p}_{22}^\top \mathbf{X}}{\mathbf{p}_{23}\mathbf{X}}\right)^2 \tag{4.53}$$

The optimization is done with the *Levenberg-Marquardt* algorithm.

## 4.9 Extending the method use to multiple views

So far we have seen how it is possible to reconstruct three-dimensional points from point correspondences on two photographs of a scene. But what happens if we have more than two photographs? It would be nice to be able to incorporate more photographs in the reconstruction of the scene, so that more points can be calculated. Actually this is most of the times necessary since with only two views many parts of the scene are occluded.

The research done on the field is in trying to generalize the epipolar constraint in the case of more that two views. This leads to constraints that can be used in the simpler and implementable case: in triplets of images. This along with a method to relate consecutive image triplets [12] can give us a way to utilize all the additional images of the scene.

This however can often be hard to implement efficiently. This is because of arithmetic inaccuracies in the implementation machines and also because of the *consecutive* nature of the solution, that can result in accumulation of error.

In this thesis we propose a different solution to the problem. Our solution has the advantage of being implementation friendly and also removes the *consecutive* nature of the solutions based on multi-view constraints. The proposed method uses the algorithm that we described previously in this chapter on pairs of images chosen from the available images of the scene. We then use point correspondences across more that two images to relate the resulting three-dimensional models and merge them into a single model. We will continue by explaining how is that possible and we will describe the algorithm to accomplice that.

### 4.9.1 Relating reconstructed segments

Suppose that we have a series of more that two photographs of a scene. We also have a number of feature points on these images. Until now we needed correspondences of these feature points between two photographs, but it is possible to have a feature point visible across more that two photographs. We will use these feature points to relate the separate reconstructions that we can obtain through the algorithm we described previously in this chapter.

Lets take three photographs from the given series. The only constraint on the choice of photographs is that there are some feature points visible across all three of the photographs. So now we have three sets of feature points:

- The set $A$ of feature points that are in correspondence in the first and second photographs
- The set $B$ of feature points that are in correspondence in the second and third photographs
- The set $C$ of feature points that are in correspondence in all three photographs

The first set $A$ will be used to reconstruct three-dimensional points from the first and second photograph, while the set $B$ will be used to reconstruct three-dimensional points from the second and third photographs. Unfortunately these reconstructed points will not be in the same world coordinate frame. This is due to the way the epipolar geometry reconstruction method works. Using the epipolar geometry to obtain the projection matrixes results to projection matrixes with unknown scale, and the first projection matrix is always the *canonical projection matrix.* We will use the feature points in the set $C$ to bring all the feature points in the same world coordinate frame.

### 4.9.2 Aligning the world coordinate frames

So for every pair of photographs, we can get from the epipolar geometry reconstruction method a reconstructed segment of the scene, but in its own world coordinate frame.

So if we have two reconstructed segments we will have to "align" the different world coordinate frames. What we basically need to find is the affine transformation that when applied on one of the segments, it will bring it in the coordinate frame of the other.

This transformation is defined by a rotation matrix $\mathbf{R}$, a scale matrix S and a translation vector $\mathbf{t}$. Our goal it to determine $(\mathbf{R}, \mathrm{S}, \mathbf{t})$. We do this by using the feature points that belong to set $C$. With these triplets of feature points we get two different reconstructed three-dimensional points. One from the first and second photograph and one from the second and third. These two points are actually the same point in three-dimensions, but the algorithm will not produce the same coordinates. So suppose that $\mathbf{x}_i$ are the reconstructed three-dimensional points from the first and second photographs, and $\mathbf{y}_i$ are the points that are reconstructed from the second and third photographs. There is an $(\mathbf{R}, \mathrm{S}, \mathrm{S}, \mathbf{t})$ that will satisfy the relation:

$$\mathbf{y}_i = \mathbf{R}\mathrm{S}\mathbf{x}_i + \mathbf{t} \tag{4.54}$$

Suppose now that we use an Euler angles parameterization for the rotation matrix $\mathbf{R}$, then we have three unknowns for the rotation matrix, three for the diagonal scale matrix, and three unknowns for the translation vector $\mathbf{t}$. When we have $n$ triplets of feature points we can solve this problem by minimizing this criterion:

$$\sum_{i=0}^{n} (\mathbf{y}_i - \mathbf{R}\mathrm{S}\mathbf{x}_i + \mathbf{t})^2 = 0 \tag{4.55}$$

This is a minimization problem that we solve using the *Levenberg-Marquardt* optimization algorithm. To be able to obtain the exact transformation we require at least three points, that are not in a degenerate form[3].

After obtaining the affine transformation $(\mathbf{R}, \mathrm{S}, \mathbf{t})$ we can use it to transform the points $\mathbf{x}_i$ in the world coordinate frame of the points $\mathbf{y}_i$ and have one "merged" reconstructed point cloud.

This procedure can be followed for other couples of paired photographs in order to reconstruct the full scene from a range of any number of photographs.

So if we have $n$ images as shown in Figure 4.10, we can reconstruct the scene using the algorithm described in Algorithm 1.

### 4.9.3 Getting more feature points

As we have seen in this chapter, given some feature points correspondences on photographs of a scene, we can reconstruct the original three-dimensional points the images where extracted from. To get a good representation of a scene we often need many

[3]like being collinear

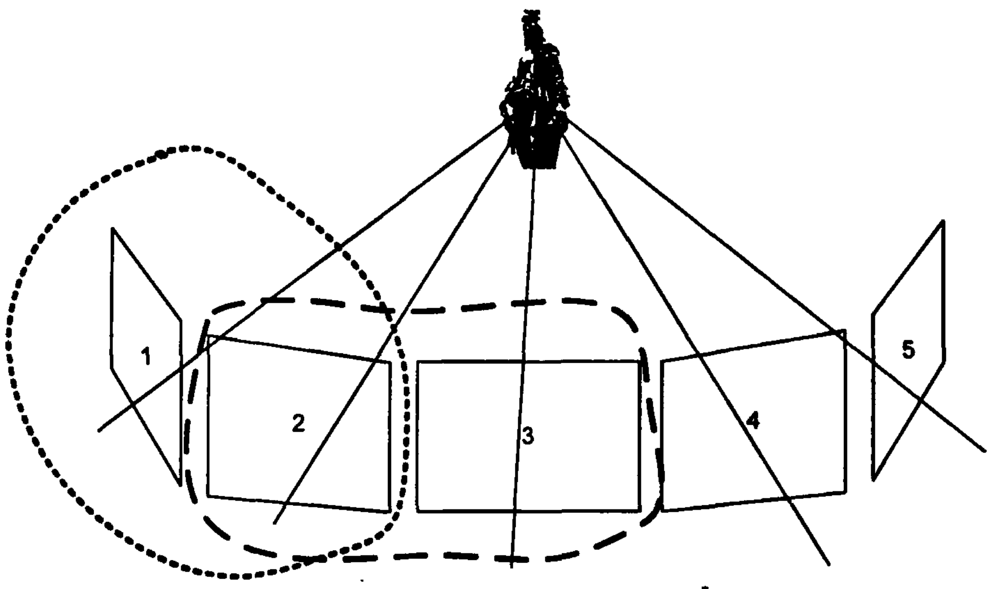


Figure 4.10:

**Algorithm 1** Multi-view reconstruction

```
Reconstruct the scene's fragment $S$ that corresponds to images 1,2
i=2
**while** $i < n$ **do**
  Reconstruct the scene's fragment $S_{new}$ that corresponds to images $i, i+1$
  Calculate the affine transform **M** that brings $S_{new}$ to the world frame of $S$
  Transform $S_{new}$ with **M**
  Merge $S_{new}$ into $S$
  $i \leftarrow i + 1$
Return $S$
```

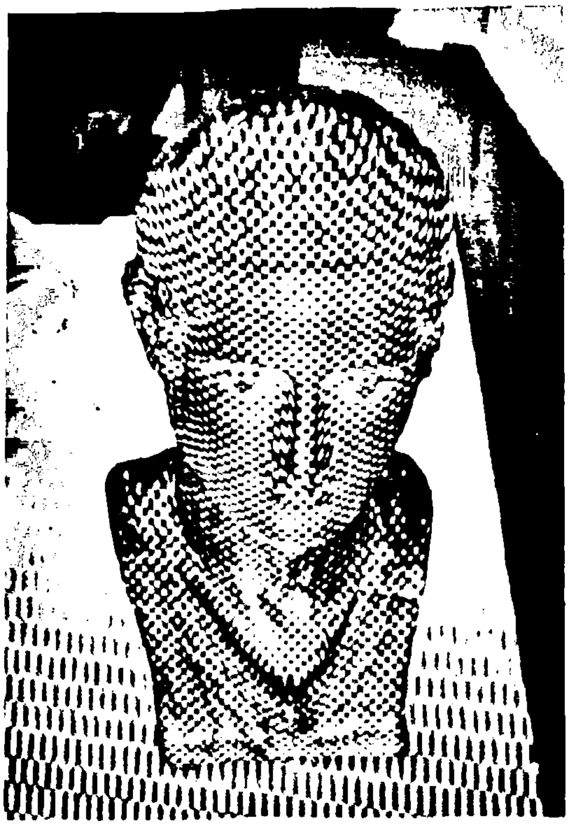


Figure 4.11: Projecting a pattern on the object we want to reconstruct creates additional feature points

feature points. The number of feature points that we can extract from a photograph depends on the scene itself. Some scenes can provide a big number of feature points while others do not give sufficient feature points to obtain a detailed reconstruction. In the case that we want to reconstruct a scene from which feature points cannot be easily extracted, we suggest the use of structured light to create additional feature points.

Structured light is light that is projected on the object and it contains patterns. For example in Figure 4.11 we can see the effect of lighting a statue with structured light. We make use of a data projector that projects on the statue a checkerboard pattern. We then take photographs of the object with the projected pattern. Note that the projected pattern must remain stable throughout the snapshot acquiring procedure.

## 4.10 Results

In this section we discuss the results that we had from the proposed method. We present the data gathered from the experimental scene reconstructions, to characterize the efficiency of the reconstruction method.

During the development of the three-dimensional reconstruction method we used several scenes to check the efficiency of the methods we were implementing. These

| Noise | Edge 2 | Edge 3 | Edge 4 |
|---|---|---|---|
| 0.00 | 1.000 | 1.000 | 1.000 |
| 0.05 | 0.999 | 0.995 | 0.998 |
| 0.10 | 0.995 | 0.992 | 0.994 |
| 0.15 | 0.982 | 0.969 | 0.978 |

Table 4.1: Results obtained from the reconstruction algorithm, from the synthesized scene. The algorithm is run several times each time adding different zero-mean gaussian noise to the on-image coordinates of the feature points. The column represent the relation of the respected edges with edge 1

scenes were both real and synthesized. We begin with synthesized images of scenes, in order to have good knowledge of the scene we were reconstructing. A synthesized scene gives us the ability to efficiently measure the performance of the implemented method.

### 4.10.1 Reconstruction of synthesized scenes

A synthesized scene is simply a scene for which we already have the three-dimensional model. From this scene we can obtain snapshots from various positions and use them for images.

So what we basically do in the case of reconstructing synthesized scenes is simulate the process of taking a photograph of a scene, and then try to reverse the process. For our experiments we used a simple synthesized scene to test the reconstruction method. This scene consists of two cubes. We then assume a camera of known intrinsic parameters **K** that we define. We position this virtual camera at different positions in space and use it to project the two cubes on "photographs". We then use the projected points on these photographs as input to our algorithm. The efficiency of the algorithm is then measured by judging the differences between the original scene and the reconstructed scene.

Since the reconstruction of the scene does not reconstruct the scene with its original scale, we cannot compare the lengths directly. What we do is check the lengths of the edges of the reconstructed cubes, and all must be of the same size. The algorithm is run several times, each time adding different amount of zero-mean gaussian noise to the screen coordinates of the feature points. We use all the available points for the calculation of the epipolar geomentry. The differences in length are shown on Table 4.1.

### 4.10.2 Reconstruction of real scenes

When the proposed method was tuned and found to efficiently reconstruct synthesized scenes, we experimented with photographs of real three-dimensional scenes. This step is necessary since it reveals the true performance of the reconstruction method.

| | X | Y |
|---|---|---|
| Focal Length | 677.573 | 679.236 |
| Principal point | 318.801 | 235.088 |
| Skew angle | 0 | |

Table 4.2: Canon IXUS v3 calibration parameters at 640 × 480

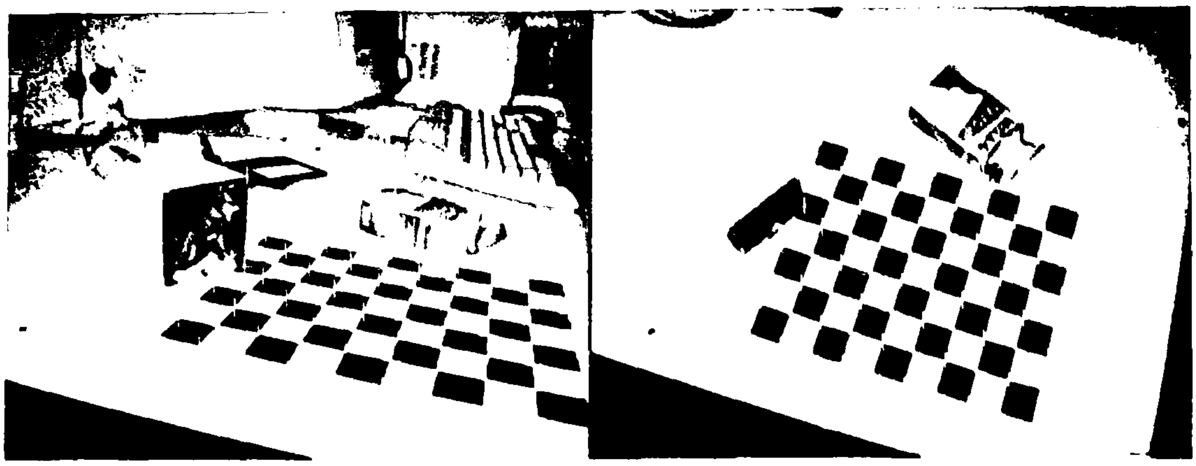


Figure 4.12: The real scene photographs used for the reconstruction

For the real scene reconstructions we used an off-the-shelve consumer camera, with unknown intrinsic parameters. The camera used is *Canon IXUS v3*. This is a *3.3 megapixel* camera that is able to take photographs up to 2048 × 1536 pixels. In our experiments we used photographs taken at 640 × 480 pixels and 1024 × 768 pixels, which was found to be efficient for our scenes.

The camera was calibrated using the method described in Section 4.2, and the intrinsic parameters where found to be the ones listed at Table 4.2.

In Figure 4.12 we can see the photographs we used for the reconstruction, with the feature points marked with white crosses. The feature points are placed by hand and not by an automated method.

The scene consists mostly of objects with good geometrical characteristics in order to be able to test these characteristics later on. The *calibration pattern* is also included in the scene for the same reason.

We test the reconstructed scene for how planar the calibration pattern is in the reconstructed scene, along with how close are some angles to 90°. For the planarity of the calibration pattern we average the distance of feature points on the calibration pattern from the plane that is defined by the three points at the edges of the calibration pattern. In the absence of noise and arithmetic computation error this distance should be zero.

We run the reconstruction algorithm each time with different feature points selected for the epipolar geometry calculation. In Table 4.3 we can see the results we obtain from the reconstruction method. The angles that we measure are marked in Figure

| Run | Angle 1 | Angle 2 | Planarity |
|---|---|---|---|
| 1 | 90.2 | 89.9 | 0.12 |
| 2 | 91.1 | 90.7 | 0.14 |
| 3 | 90.1 | 90.1 | 0.09 |
| 4 | 88.7 | 90.0 | 0.79 |
| 5 | 89.1 | 89.9 | 0.11 |
| 6 | 90.1 | 90.0 | 0.10 |

Table 4.3: Results obtained from the reconstruction algorithm. The algorithm is run several times each time choosing a different set of feature points for the calculation of the epipolar geometry

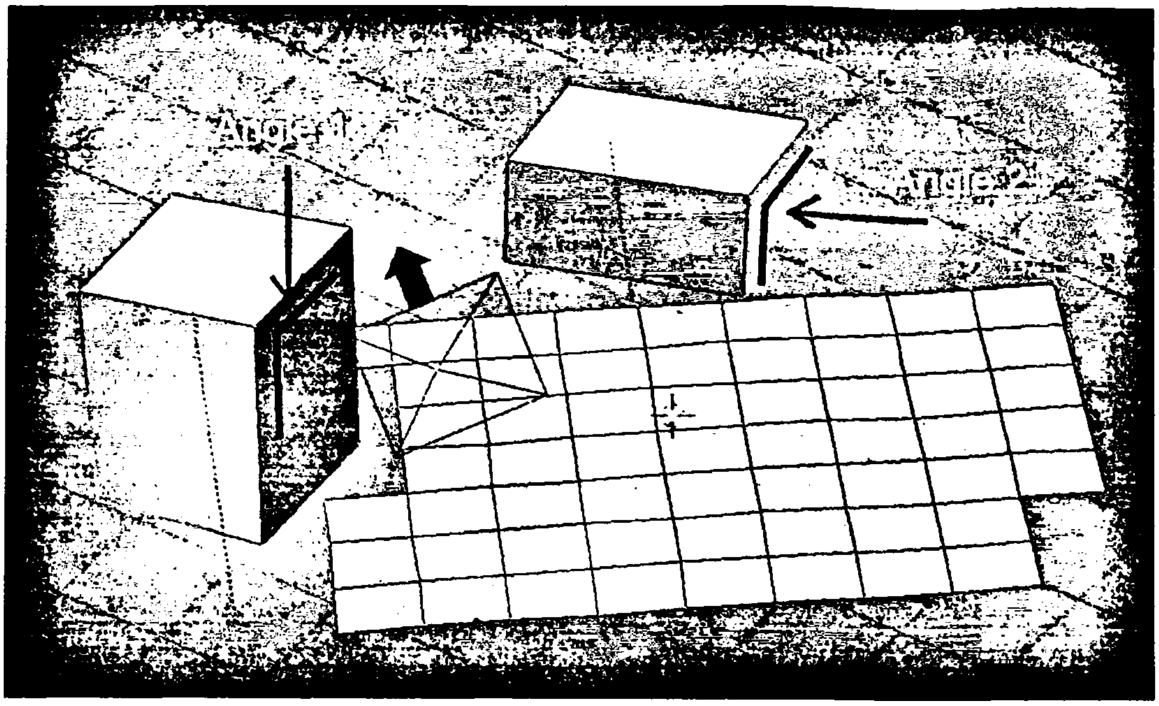


Figure 4.13: The measured angles in the reconstructed scene

4.13.

As we can see the results vary depending on the feature points that are selected for the calculation of the epipolar geometry. This is expected since the error on the image coordinates of each feature point is different. This also shows that the selection of appropriate feature points for the calculation of the epipolar geometry if crucial for obtaining accurate results from the method.

# Chapter 5

# Implementation

## 5.1 Overview

In this chapter we discuss the implementation details of the system we created in order to reconstruct three-dimensional models from photographs. We will examine the various problems that one can face in implementing the theoretical methods described in the previous chapters, along with the proposed solutions and workarounds that were necessary. We will demonstrate this way the feasibility of the effort and the prospects that we have, given the performance and accuracy of the implementation.

## 5.2 Programming language

The language of choice for the implementation of the system was a very sensitive decision that had to be made from the beginning of the project. The main requirements from the language were:

|  | Code Speed | GUI | Scientific libraries | Prototyping |
|---|---|---|---|---|
| C | Excellent | GTK | LAPACK | Bad |
| C++ | Excellent | wxWidgets | LAPACK | Bad |
| Delphi | Medium | Delphi | Very little support | Bad |
| Perl | Medium | Several | LAPACK ports mainly | Good |
| Python | Medium | Several | SciPy | Excellent |
| Matlab | Low | Native | Excellent | Excellent |

Table 5.1: Language features

- Speed of execution
- Graphical user interface creation
- Support for accurate mathematical computations
- Availability of scientific libraries
- Allow for quick prototyping

Several languages were evaluated for the purpose, some of which are:

- C
- C++
- Delphi
- Perl
- Python
- Matlab

Most of the languages reviewed usually failed in to more that one of our basic requirements, with the most mutual exclusive to be speed of execution and quick prototyping. Both of which are fundamental for the project. Speed of execution is crucial for this kind of application since if involves heavy arithmetic computations. On the other hand the time line for the project was short and there wan't a clear road-map for the implementation. So we had to create a system with no way on knowing forehand what exactly it will take, or how exactly it will be structured, and what pitfalls we were going to face. This requires that the language will support rapid prototyping, so that we could try out ideas as fast as possible.

In table (5.1) we can see the features every language provides. The columns are the features and the rows are the languages. Every language is examined for code speed, graphical user interface libraries and builders, the availability of scientific libraries, and finally the quick prototyping allowed by the language. Note that on features like graphical user interface and scientific libraries, there are usually several available choices. We base our report on these features according to the best choice among them.

It is obvious from the table (5.1) that the choice of language for the particular project is difficult. The languages that support good prototyping lack in speed and vice versa. The language Delphi which is basically an object oriented Pascal with very good graphical user interface capabilities is rejected since the support for scientific libraries is poor. Perl on the other hand is rejected because of its poor scientific libraries and because of its *general purpose* nature that is more appropriate for quick-small programs. Matlab would be an excellent choice for the purpose. The scientific computing support from matlab is excellent, since it is what it was made of. Matlab is also very good at rapid prototyping of ideas and solutions. Unfortunately the speed is not so good as C++ for example, but that is not the basic problem. The basic problem is graphical user interface and the capability of creating autonomous applications. Our purpose was to create an concrete application that does not relay on an application like Matlab to run. This forbids the use of Matlab in the project, even it would save us a lot of time with its excellent scientific features.

C on the other hand has great speed of execution but is a very low level language that is certainly not good for our purpose. The best available graphical user interface toolkit for C is GTK [17]. The fact that graphical user interfaces are from nature object oriented and C is not an object oriented languge, forces the GTK toolkit to create an object oriented model on top of C, which is very clumsy and error prune. The availability of scientific libraries for C is good but very clattered. There are several libraries that do different or the same things, some better than the others, and it is quite hard to find the best solution. So C is rejected too, after all it is a very aged language and it is basically suited for operating systems programming.

The same things goes for C++ also, with the difference that it is a fully object oriented language. The graphical user interface is very good with *wxWidgets* [18] being a very good , and portable across platforms, toolkit. The scientific libraries are mostly the same that C uses. The bad thing about C++ is that it is not suited for rapid prototyping.

A very good choice for a language is Python [19]. Python is a fully object oriented language. It is open source and provides a wide variety of libraries. Python is a very high level language that allows for quick prototyping and testing. Its dynamic nature allows to easily create generic *"templetized"* code that is very useful in quickly sketching implementations. Python also comes with a very powerful scientific library SciPy [20]. A very good feature that Python has is the ability to be extended in C of C++. The programmer can easily move code back and forth from Python to C++. This allows to quickly prototype code in Python and later move code to C++ and enjoy the execution speed of C++.

So is was decided that the development system is going to be based in a mix of the languages Python and C++. We basically worked in Python and then later moved the execution speed critical parts to C++. In retrospect this was a wise choice since it cut down the development time a lot, with no compromises in execution speed of the final application.

## 5.3 Additional software

Through the development of the reconstruction system we used some additional software. This software is in the form of libraries for Python and C++ and some stand-alone software.

Great help came from the use of Maple [21]. Maple's excellent analytical derivatives calculation capabilities allowed us to calculate the analytical derivatives of very complex functions fast and error-free.

The Python library SciPy [20] was used for the many linear algebra problems that we faced in this thesis. SciPy is a scientific library that gives Python the ease of use and features of a suite like Matlab [22]. It allows easy matrix manipulation and many linear algebra algorithms, along with some optimization methods.

The mesh modelling application Blender [23] was also used for triangulating and presenting the resulting point clouds that our application generated.

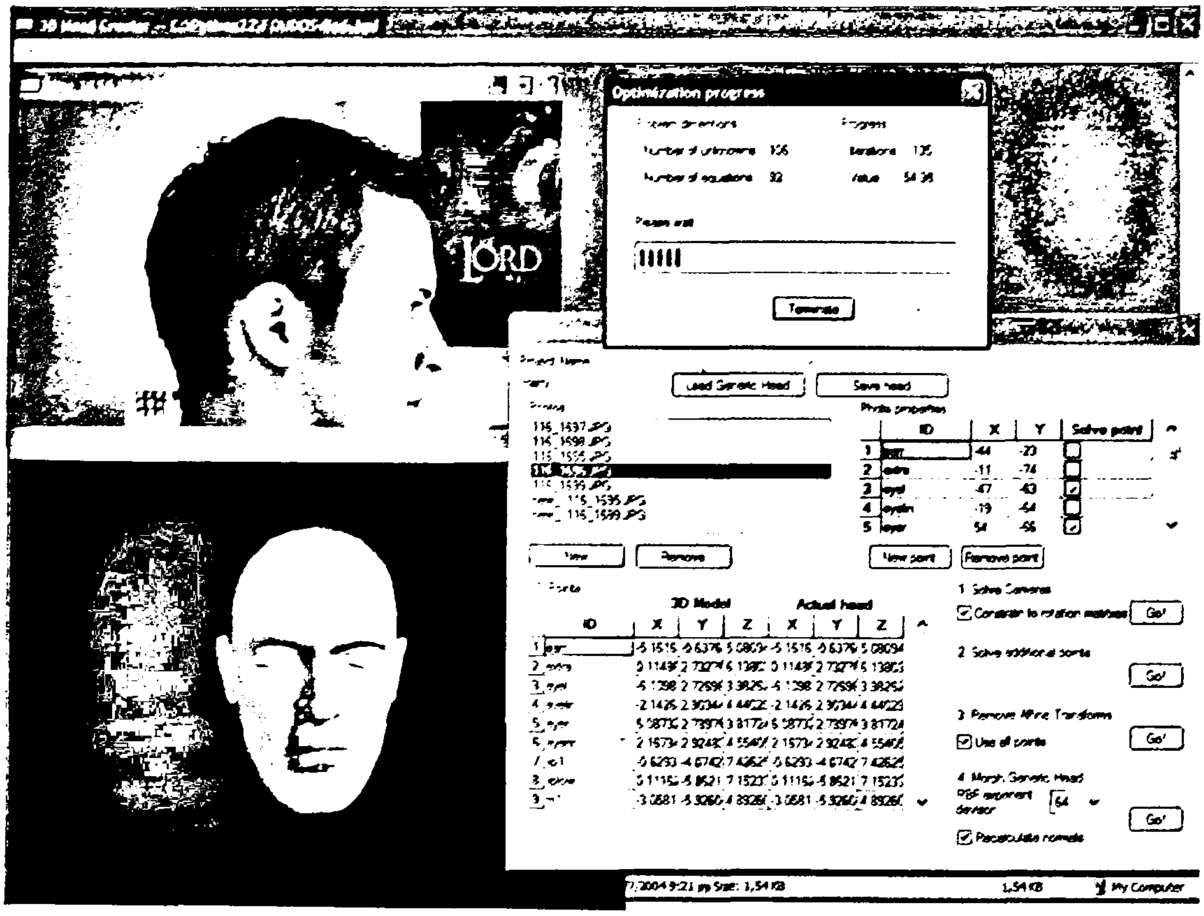
## 5.4 Simultaneous structure and motion method

We first tried the implementation of the method described in Section 3.2, that enables us to recover structure and motion simultaneously.

For the parameterization of the rotation matrixes we used the parameterization shown at equation (3.4).

The implementation of this method didn't result in good results. The optimization algorithm turned out not to able to find the minima of the problem, and usually stuck in local minima. The problem begins with the fact that the parameterization of the rotation matrix requires trigonometric functions that are hard to manipulate with gradient based optimization methods. Specially when the functions are so complex.

In the beginning of the testing and to estimate if the results were going to be good, we used forward differencing for calculating of the Jacobian matrix passed to the *Levenberg-Marquardt* optimization algorithm. The results was not good. First of all the speed of execution was very low, since the function had to be evaluated many times to estimate the Jacobian. The function is also slow by itself since it contains a lot of trigonometrical functions, and can be very complex for big problems with many views and point matches. Speed was not the only issue though. The converge to slow and



Problem dimensions

Number of unknowns 106

Number of equations 32

Progress

Iterations 135

Value 54.38

Please wait

Photos

116_1597.JPG
116_1598.JPG
115_1595.JPG
116_1596.JPG
115_1599.JPG
new_115_1595.JPG
new_116_1599.JPG

Photo properties

| | ID | X | Y | Solve point |
|---|---|---|---|---|
| 1 | earr | -44 | -23 | ☐ |
| 2 | ebro | -11 | -74 | ☐ |
| 3 | eyel | -47 | -43 | ✓ |
| 4 | eyebrl | -19 | -64 | ☐ |
| 5 | eyer | 54 | -56 | ✓ |

Points

| | | 3D Model | | | Actual head | | |
|---|---|---|---|---|---|---|---|
| | ID | X | Y | Z | X | Y | Z |
| 1 | earr | -5.1515 | -9.6376 | 5.0809 | -5.1516 | -9.6376 | 5.08094 |
| 2 | ebro | 0.1143 | 2.7327 | 6.1380 | 0.1143 | 2.7327 | 6.13803 |
| 3 | eyel | -5.1398 | 2.7259 | 3.3825 | -5.1398 | 2.7259 | 3.38253 |
| 4 | eyebrl | -2.1426 | 2.3034 | 4.4422 | -2.1426 | 2.3034 | 4.44223 |
| 5 | eyer | 5.3873 | 2.7397 | 3.8172 | 5.3873 | 2.7397 | 3.81724 |
| 6 | eyebrr | 2.1573 | 2.9243 | 4.5540 | 2.1573 | 2.9243 | 4.55402 |
| 7 | ml | -5.6293 | -4.6742 | 7.4262 | -5.6293 | -4.6742 | 7.42625 |
| 8 | mr | 5.1115 | -5.8621 | 7.1523 | 5.1115 | -5.8621 | 7.15235 |
| 9 | [illegible] | -3.0581 | -5.3260 | 4.8926 | -3.0581 | -5.3260 | 4.89260 |

1 Solve Cameras

[✓] Constrain to rotation matrices

2 Solve additional points

3 Remove Affine Transform

[✓] Use all points

4 Morph Generic Head

RBF exponent divisor 64

[✓] Recalculate normals

7/2004 9:21 pp Size: 1,54 KB 1,54 KB

Figure 5.1: The *3D Head Creator* application that uses the simultaneous structure and motion method successfully

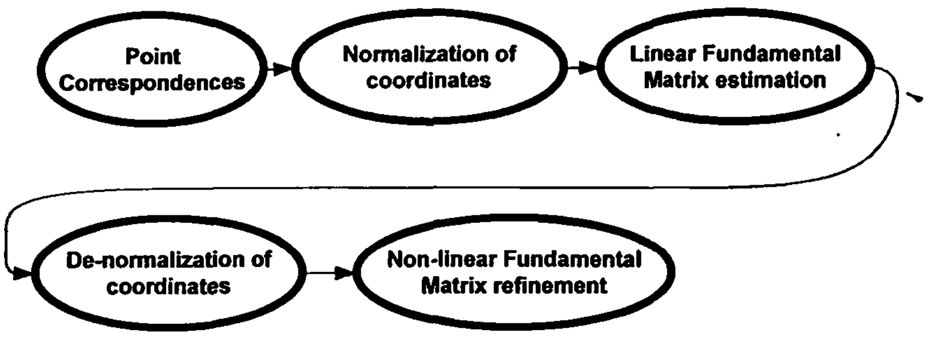

Point Correspondences
Normalization of coordinates
Linear Fundamental Matrix estimation
De-normalization of coordinates
Non-linear Fundamental Matrix refinement

Figure 5.2: The procedure followed in order to efficiently estimate the epipolar geometry

the method got stuck in local minima most of the times.

We suspected that the problem was the inaccuracy of the estimated Jacobian matrix. So we implemented a system to analytically derive the Jacobian for the function to be solved. It turned out that the method performed worst with the analytical Jacobian. Of course the execution speed was much higher with the analytical Jacobian, but the method got stuck at local minima much often now. It seems that the trigonometrical functions created *high frequency* disturbances in the function, that the inaccuracy of the estimated Jacobian helped to skip over, finding better minima.

By initialing the method several times with random starting points we were able to find the global minima, but the procedure is slow and does not guaranty results. So this method was judged inappropriate for the purpose of our application. The implementation is there and the user can try it in the application, but this is not the suggested method for reconstruction. The user should use the *Epipolar geometry method* most of the times.

This method, even if it is not appropriate for the general reconstruction system we were trying to implement, does not mean that is useless in general. For example we were able to use this method successfully in a human head reconstruction system (Figure 5.1). In this system we had the advantage of knowing the general shape of our object. This way the application could feed the optimization algorithm with a good starting point using the coordinates of points on a *generic head.* The solution for the specific head the user is working on, can't be far from the generic head so the optimization system find the solution easily.

## 5.5 The epipolar geometry method

The implementation of the reconstruction method that uses the epipolar geometry described in chapter 4, was the most successful. The results obtained from this method are very satisfactory.

### 5.5.1 Estimating the fundamental matrix

The first part of this method requires the estimation of the epipolar geometry between the two views. This is equivalent to estimating the fundamental matrix , as described in Section 4.4.

During the development of the fundamental matrix estimation system we used many different approaches. The first was the eight-point algorithm described in section 4.4.1 (page 26). The algorithm is very straightforward and requires solving a set of linear equations, which is quick and always gives a solution. The implementation is actually just doing a *Singular Value Decomposition*.

However this method does not perform well under noisy real data. It specially fails to enforce the rank 2 constraint required for the fundamental matrix. This is solved by taking the closest rank 2 matrix to the one the eight-point algorithm provides. But again the eight-point algorithm does not return appropriate results for the fundamental matrix [6].

So we went on to implementing the method that uses a non-linear criterion and described in Section 4.4.3. The system uses the rank enforcing parameterization of the fundamental matrix as shown below:

$$\mathbf{F} = \begin{bmatrix} a & b & -ax - by \\ c & d & -cx - dy \\ -ax' - cy' & -bx' - dy' & (ax + by)x' + (cx + dy)y' \end{bmatrix} \tag{5.1}$$

The optimizer for the non-linear criterion was first implemented in Python and used the Levenberg-Marquardt non-linear least squares optimization algorithm as implemented by SciPy [20]. The results of the algorithm was satisfactory and it gives a good approximation of the fundamental matrix. Actually the results are much better than the results obtained by the linear eight-point algorithm. However this method possesses the defects of all non-linear gradient based optimization methods, as it depends on a good starting point. In the beginning we used random initialization for the algorithm, which often got stuck in local minima.

To overcome this problem we used the previously implemented eight-point algorithm to estimate a good starting point to feed in the non-linear optimization step. This actually worked efficiently and the non-linear optimization algorithm did not stuck in local minima most of the times. When it does a re-run of the algorithm, with a slightly jittered starting point, will find the correct solution.

Even though with the coordinate normalization, the linear eight-point algorithm performs almost as good as the non-linear one, we still use the non-linear optimization step to further improve the result.

The non-linear fundamental estimation step was also rewritten in C++ to get the maximum execution speed possible. That with the good initial guess for the solution coming from the eight-point algorithm we get instant estimation of the epipolar geometry at the bigger accuracy possible.

For the non-linear optimization step, the analytical Jacobian is provided to the optimizer to speed up the optimization's converge. The analytical partial derivatives were calculated with the aid of Maple [21].

In Figure 5.2 we can see the full epipolar geometry estimation process.

### 5.5.2 Computing camera's motion and scene structure

Once the fundamental matrix is known the application can continue to derive the motion of the camera. This procedure was described in Chapter 4. We saw that it is possible to obtain the cameras' projection matrixes when the fundamental, or the essential matrix is known.

The application calculates the four possible projection matrices by evaluating the equations (4.47). It then uses the *"points in front of the camera"* (Section 4.8.3) criterion to choose the correct projection matrix for the second camera.

Structure is then calculated with triangulation, as described in Section 4.8.4.

## 5.6 The graphical user interface

The application that we created uses a graphical user interface to interact with the user. This graphical user interface was created with the library wxPython, which is basically a wrapper around wxWidgets for C++. This library allow us to make the application portable across platforms. This means that the application can run in a variety of platforms, from Windows to many Unix clones with X-Windows based graphical environments.

In Figure 5.3 we can see how the main graphical user interface looks while running on Windows XP. The application follows the *Multiple Documents Interface(MDI)* paradigm, with documents being images and reconstruction views. The user can create and work on one project at a time. This project consists of the images that we took from a scene along with the image points correspondences. These projects can be saved to spacial files with the extension *.mpf*, which stands for *modelling project file.* All required information about the project is saved in these files and can be loaded at a later time.

The bottom and right part of the graphical user interface are adjustable in size and contain several tab pages. Each tab page has a special purpose. At the bottom part the tab pages are the "Info" and "Images". The "Info" tab contains general information about the currently open project. This info is general to the project, like the name of the project, the number of images used in the project, etc. The tab labelled "Images"

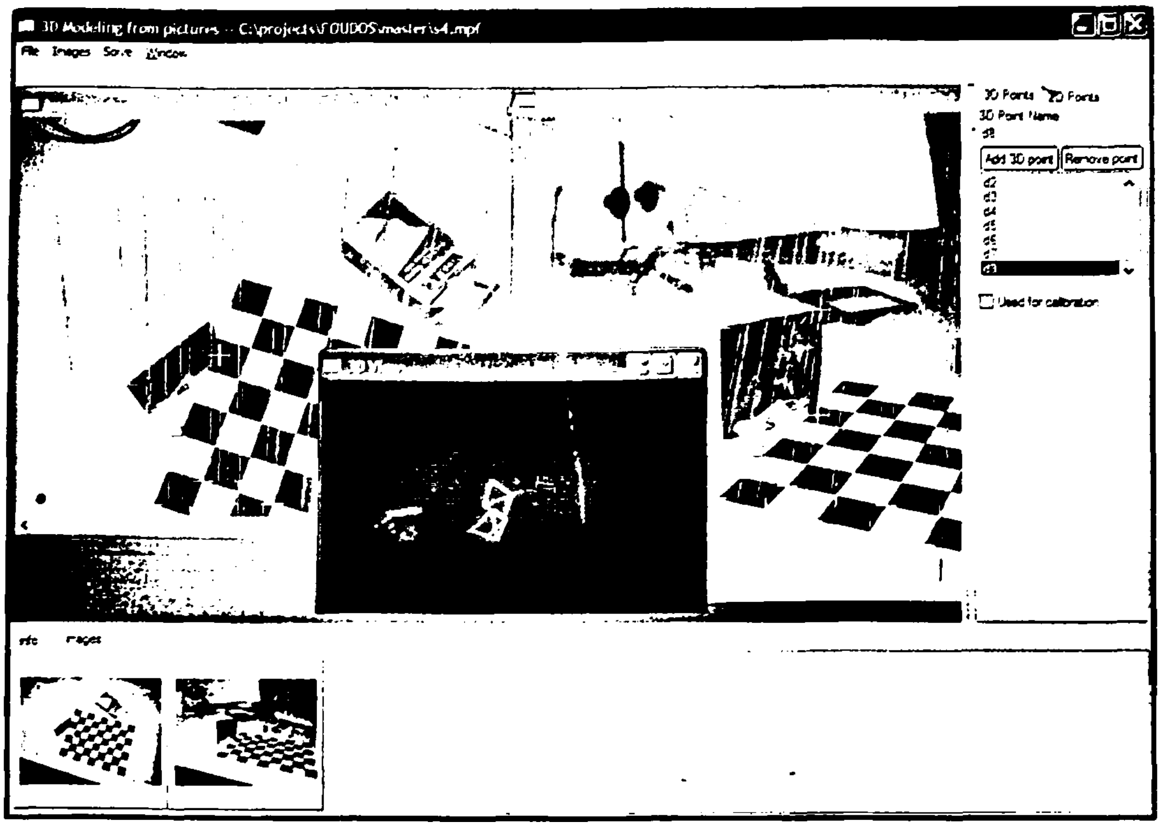


Figure 5.3: The main graphical user interface of the application. In the screen-shot we can see an open reconstruction project. Two images of the scene are visible along with the calculated corresponding epipolar lines. The three-dimensional reconstruction is also rendered in the OpenGL window.

(Figure 5.4), contains the thumbnails of all the images in the currently open project. The user can quickly select the image that interests him from this toolbar.

The tabs at the right part (Figure 5.5) of the graphical user interface are dedicated to controlling the point placement and correspondence.

The menu at the top of the application allows the user to take actions concerning the modelling project. These range from loading and saving the project file, to solving the problem of reconstruction with one of the available methods.

### 5.6.1 Global refinement step

The method that uses the epipolar geometry to reconstruct the scene can be further refined with the use of the global optimization step from the simultaneous structure and motion method.

When the structure and motion of the scene has been estimated with the epipolar geometry method, a maximum likelihood estimation can be obtained through *bundle adjustment* [13]. What we do is refine the projection matrixes $\mathbf{P}_k$ and the three-dimensional points $\mathbf{X}$ that we obtained with triangulation, so that the mean squared

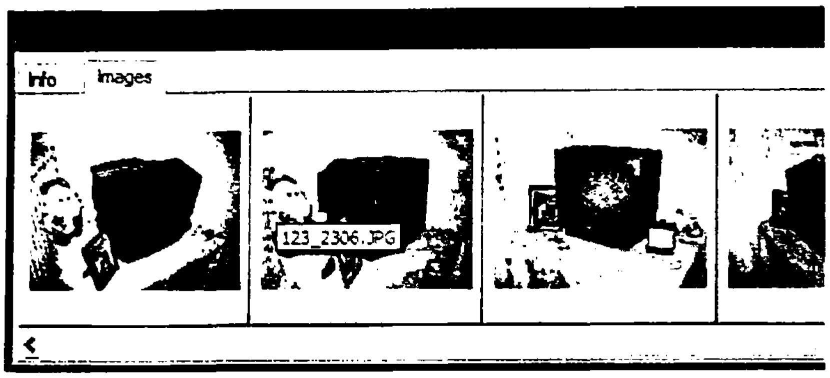


Figure 5.4: The image selection toolbar with thumbnail previews of the images in the project file.

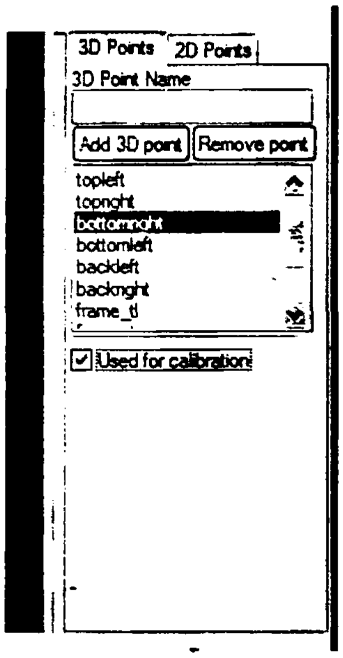


Figure 5.5: The right part of the graphical user interface that allows for point manipulation.

distances between the observed image points $\mathbf{x}_{ki}$ and the reprojected ones, is minimized. This is basically what we do in the simultaneous structure and motion method. The following criterion is minimized:

$$\min_{\mathbf{P}_k \mathbf{X}_i} \sum_{k=1}^{m} \sum_{i=1}^{n} D(\mathbf{x}_{ki}, \mathbf{P}_k \mathbf{X}_i)^2 \tag{5.2}$$

where $D(\mathbf{a}, \mathbf{b})$ is the Euclidian image space distance. If the image error is zero-mean Gaussian then bundle adjustment is the maximum likelihood estimator. This way we bootstrap the Levenberg-Marquardt optimization algorithm with a starting point very close to the solution and the algorithm converges quickly to a more refined solution.

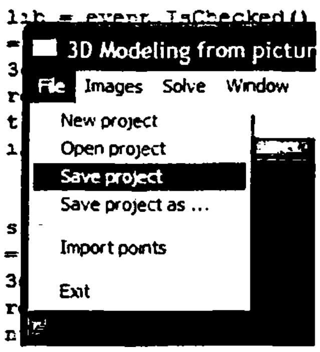


Figure 5.6: The file menu of the application.

### 5.6.2 Creating a new project

To reconstruct a three-dimensional scene from photographs using this application the user must first create a new modelling project. This is done by selecting the "new project" item of the "File" menu. A new empty project is the created and the user should continue to add images to it.

### 5.6.3 Adding images

Then the user should import a number of photographs in the application. The application supports most of the available standard image formats, like JPEG, PNG, GIF, BMP, etc. In general it is recommended that the JPEG format is used for photographs. This is also true for this application, but for one additional reason too. The JPEG files when taken from digital cameras can contain additional info about the camera that took the picture. Some of this info, like the camera's focal length, are very useful to our application and the JPEG file format is the only that supports it.

To add a new photograph the use should use the "Add images" item of the "Images" menu. A file selection dialog will appear that will prompt the user to select the photograph from the disk. The new photograph will be inserted in the current project and should be visible in the thumbnail toolbar (Figure 5.4) as a thumbnail. The user should repeat this operation an many times as necessary to add all the available photographs of the scene.

### 5.6.4 Adding points

Once the photographs are loaded to the application, the user will have to establish point correspondences among the images. This is done by first adding named three-dimensional points to the project. At the "3D Points" tab page (Figure 5.5) the user can add three-dimensional points by typing the name of the point and pressing the "Add 3D point" button. Note that we enter here the points that we want the application to reconstruct. Some of these points will be used for the calibration of the scene also.

The user should mark these point by checking the "Used for calibration" button. The points that are not used for calibration will still be reconstructed.

Once the points are entered in the application the user should select these points from the list and click on the images that the point is visible. Crosses will mark the position of the point in each image. The currently selected point has a red cross while the others a white cross.

The user should use for calibration the points that are better placed. This way the overall process will be more accurate. In the same sense the user would not use for calibration a point that has big ambiguity in its position. Such a point is fine for being reconstructed but putting it in the calibration procedure will tain the results of the other points too.

### 5.6.5 Solving the scene

The application allows for two distinct methods of solving the reconstruction problem. The one that was described in Chapter 3 and the one described in Chapter 4.

The first method uses the one big optimization step that was described at the according chapter. This method is there for illustration purposes only and should not be used for actual reconstruction of three-dimensional scenes. This method fails most of the times to reconstruct complex scenes. But is can sometimes give good results, depending on the initialization which can be no better than random. This scene solving is selected by choosing the "Simultaneous solution of motion and structure" item of the "Solve" menu.

The proposed way to reconstruct a scene is to use the "Estimate Fundamental matrix" item from the "Solve" menu. This solution utilizes the theory of chapter 4 to reconstruct the three-dimensional scene. This method is accurate and produces the desirable results. When we select this method there must be two photographs open in the application. Then the program will try to estimate the epipolar geometry of the two views. In Figure 5.7 we can see the result of the epipolar geometry calculation by the program.

We can observe the epipolar lines passing from the corresponding point in the images. Once the epipolar geometry is estimated we can calculate the structure of the scene and the motion of the cameras. This is done automatically when the user chooses to estimate the epipolar geometry, but can be done manually by choosing the "Generate 3D Points" from the "Solve" menu. This is useful for when the user makes changes to the point on the two images but does not want to recalculate the epipolar geometry. This can happen when modifying points not used in the calibration and epipolar geometry calculation step, or if the user would like to see how a point will be reconstructed if it had different image space coordinated, with the same epipolar geometry.

The resulting three-dimensional reconstruction can be viewed from the embedded

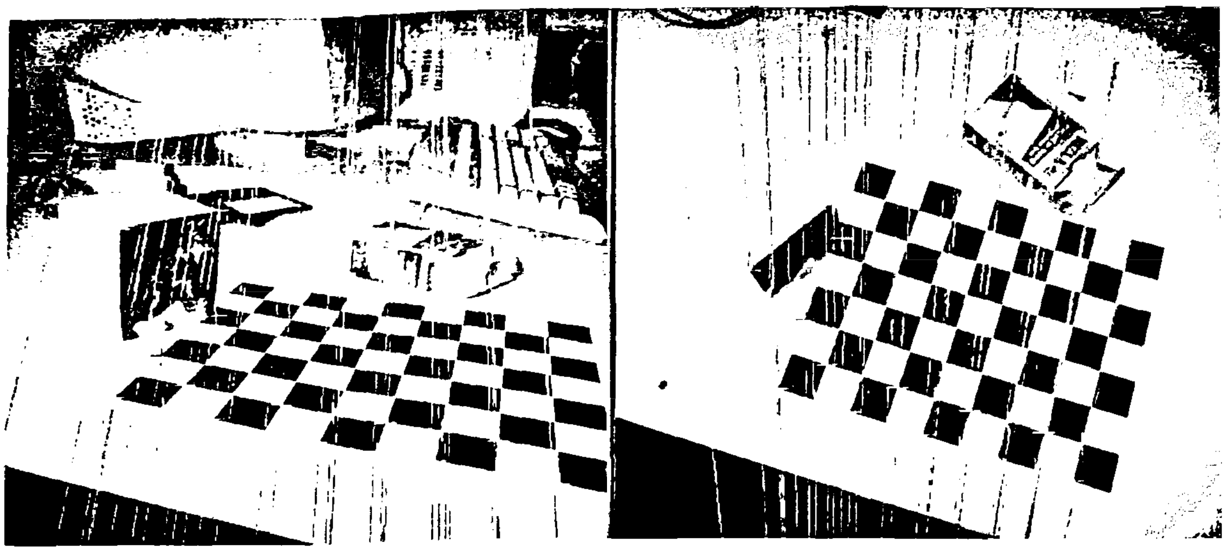


Figure 5.7: The epipolar lines of the two views

"3D View" window, or can be exported to a *.ply* file. These files are three-dimensional mesh files that can be read by many modelers. This file is mainly used by the *Stanford 3D scanning repository*, and there are importers available for many modelling programs. This way the resulting cloud point can be imported in a modelling suit for further retouching it.

In Figure 5.8 we can see the reconstructed model in the modelling application : Blender [23]. The point cloud was exported from our application in the *.ply* file format and imported easily to Blender, where we were able to triangulate it. As it can be seen the quality of the reconstruction is very good. In Figure 5.9 we can also see the model illuminated and shaded.

## 5.7 Example reconstruction

The reconstruction system we developed was used to reconstruct the bush of the "Charioteer" model. The replica bust of the famous Charioteer can be seen in Figure 5.10. To efficiently reconstruct the model we used structured light. This was necessary since the model does not provide many feature points by itself. In Figure 5.11 we can see the two photographs that we actually used for the reconstruction. The projected pattern on the bust enabled us to extract 250 feature point correspondences. These feature points was marked in the reconstruction application and the result was the three-dimensional positions of the feature points. In the specific model we didn't have to reconstruct both sides of the model. By exploiting the symmetry of the model there is no need to extract feature points from the other side of the face. We reconstructed one side of the model and then mirrored these points to obtain the other side.

The resulting point cloud was exported from our application and imported in the

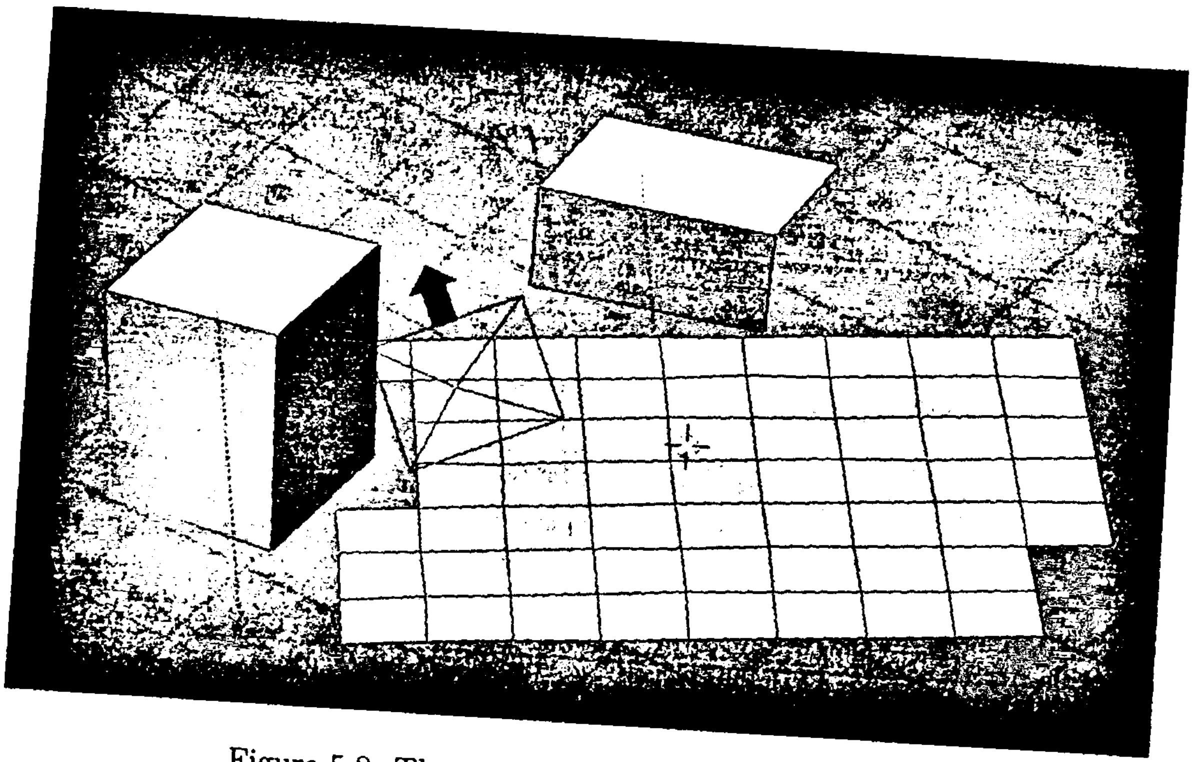


Figure 5.8: The reconstructed wire-frame model

modelling application *Blender* where is was triangulated and later textured. The resulting model can be seen in Figure 5.12.

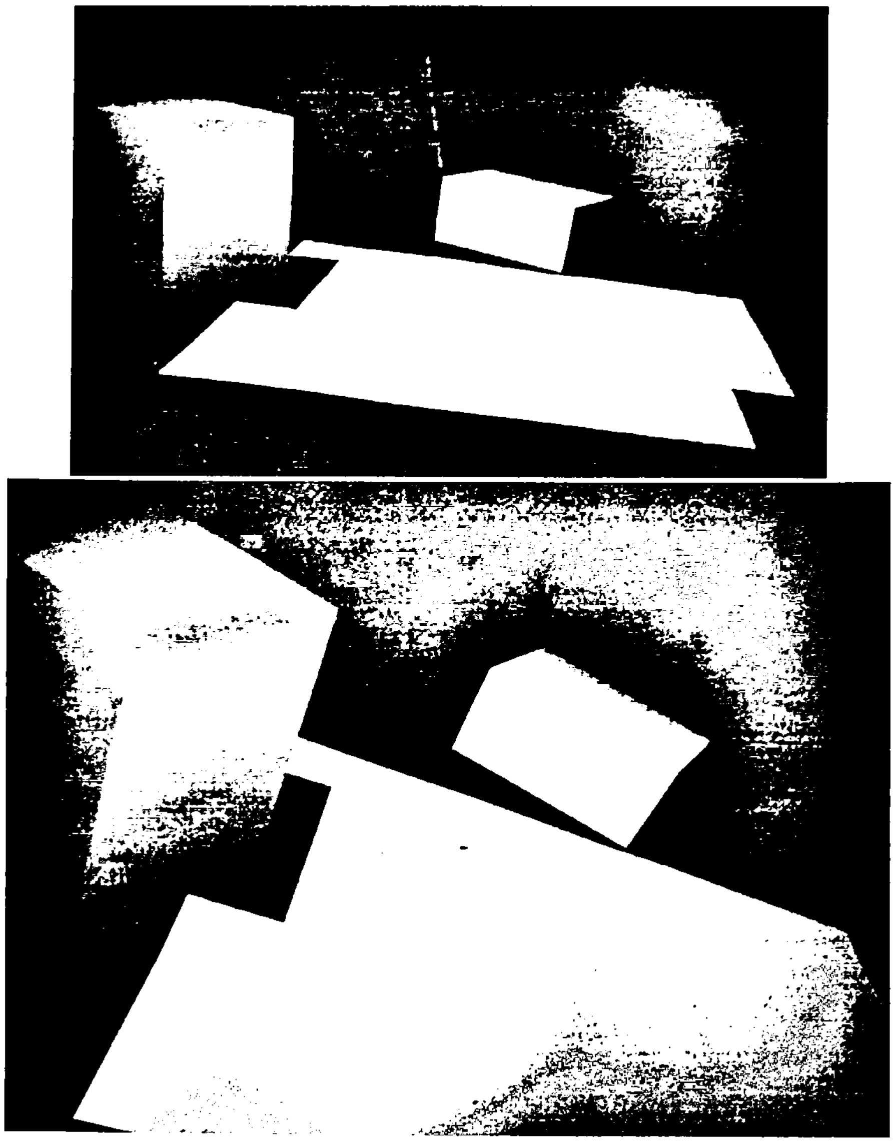


Figure 5.9: Two views of the reconstructed model rendered with shading

Figure 5.10: The replica bust of *Charioteer*, we are going to reconstruct

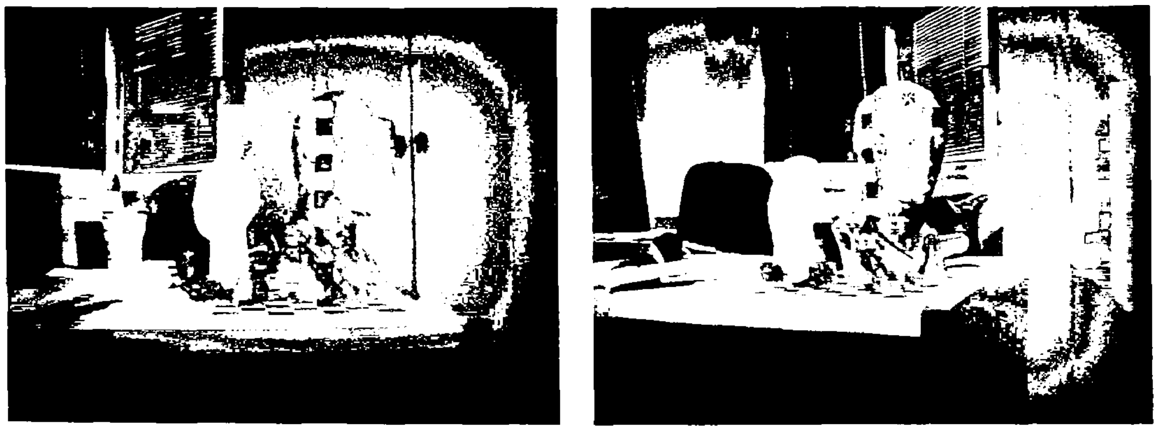


Figure 5.11: The photographs used for the reconstruction. To extract a fine grid of feature points a pattern was projected on the bust. The result are 250 feature point correspondences

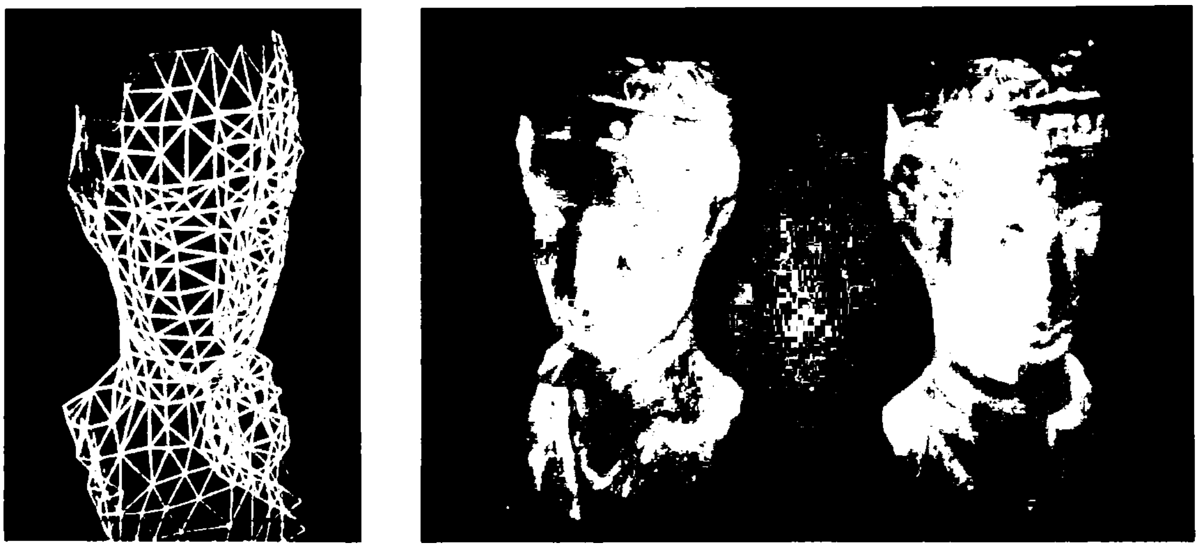


Figure 5.12: The reconstructed model. In the left frame we can see the wireframe model. In the right we see two views of the model with a texture from the original bust applied

# Chapter 6

# Conclusions and future work

The work presented in this thesis deals with the three-dimensional reconstruction of scenes from two-dimensional snapshots. The reconstruction problem was studied from several perspectives and a solid procedure was developed around which an interactive reconstruction application was build.

The developed method is decomposed into a number of discreet tasks. This design allowed for easy testing of different algorithms for each task. We where also able to identify bottlenecks and pitfalls in the total reconstruction system, and to optimize the performance of the system both speed wise but also in terms of accuracy.

The first system we developed that was based around a simultaneous reconstruction of structure and motion was found inefficient for the purposes of our application. The optimization step involved made the system inappropriate for cases that no good starting point for the solution is known. However this method waS useful in our reconstruction system, as a final refinement step.

The proposed algorithm for reconstruction in this thesis is based on the *epipolar geometry* between two views. We described a method that first estimates the epipolar geometry of two views, to calculate the motion of the cameras. Then with the aid of triangulation we are able to reconstruct actual three-dimensional points of the scene, from point correspondences. Various methods are described that optimize the output of the algorithms to make it usable in practice.

We have contributed a method to extend the usage of the epipolar geometry method to take advantage of more than two photographs. The proposed method does not relay on constraints of three view (or more) geometry, but works using data available only from the epipolar geometry method.

This work depends heavily in the extraction of good feature points from the photographs. This procedure can be performed interactively but it can also be automated. An automated method is going to be implemented for the reconstruction system based on detecting feature points with image processing and then later use the epipolar geometry to detect point correspondences.

# BIBLIOGRAPHY

[1] Richard Szeliski and Sing Bing Kang. Recovering 3D Shape and Motion from Image Streams using Nonlinear Least Squares, *Journal of Visual Communication and Image Representation*, 5(1):10 – 28, March 1994.

[2] O. Faugeras, What can be seen in three dimensions with an uncalibrated stereo rig, *Computer Vision - ECCV"92, Lecture Notes in Computer Science*, Vol. 588, Springer-Verlag, pp. 563–578, 1992.

[3] R. Hartley, Estimation of relative camera positions for uncalibrated cameras, *Computer Vision - ECCV"92, Lecture Notes in Computer Science*, Vol. 588, Springer-Verlag, pp. 579–587, 1992.

[4] P. Torr, P. Beardsley and D. Murray, Robust Vision, *Proc. British Machine Vision Conference*, 1994.

[5] R.Y. Tsai and T.S. Huang, Uniqueness and estimation of three dimensional motion parameters of rigid objects with curved surfaces. *IEEE Trans. Patt. Anal. Machine Intell.* PAMI-5:13–27, 1984.

[6] Q.-T. Luong, R. Deriche, O. Faugeras, and T. Papadopoulo, On determining the fundamental matrix: Analysis of different methods and experimental results. *Technical Report RR-1894, INRIA*, Sophia Antipolis, France, 1993

[7] A. Shashua and S. Avidan, The Rank-4 Constraint in Multiple View Geometry. In *Proc. of ECCV'96*, volume 2, pages 196–206, 1996

[8] S. Avidan and A. Shashua, Tensor embedding of the Fundamental Matrix, In *Proc. of post-ECCV SMILE'98*, volume Springer LNCS 1506, pages 47–62, 1998

[9] R. I. Hartley, In defence of the 8-point algorithm, In *Proc. of the 5th ICCV'95*, 1995

[10] Z. Zhang, A Flexible New Technique for Camera Calibration. *Technical Report MSR-TR-98-71*, Microsoft Research, December 1998.

[11] Z. Zhang, Flexible Camera Calibration by Viewing a Plane from Unknown Orientations. *In Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 666–673, September 1999.

[12] M. Lourakis and A. Argyros, Efficient 3D Camera Matchmoving Using Markerless, Segmentation-Free Plane Tracking. *FORTH-ICS*, 2003

[13] D. Brown, The bundle adjustment - progress and prospect, *XIII Congress of the ISPRS*, Helsinki, 1976.

[14] Z. Zhang and C. Loop, Estimating the Fundamental Matrix by Transforming Image Points in Projective Space. *Computer Vision and Image Undestanding*, Vol.82, No.2, pages 174–180, May 2001

[15] Z. Zhang, R. Deriche, O. Faugeras, Q. Luong, A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry. *Robotique, image et vision*, May 1994

[16] Z. Zhang, Determining the Epipolar Geometry and its Uncertainty: A Review. *The international Journal of Computer Vision*, 27(2):161–195, March 1998. Also Research Report No.2927, INRIA Sophia-Antipolis.

[17] *http://www.gtk.org/*

[18] *http://www.wxWidgets.org/*

[19] *http://www.python.org/*

[20] *http://www.scipy.org/*

[21] *http://www.maplesoft.com/*

[22] *http://www.mathworks.com/*

[23] *http://www.blender.org/*

# INDEX