An Application of Photogrammetry in the Petrochemical Industry

by

Wynand Singels

Thesis presented in partial fulfilment of the requirements for the degree of Master of Science in Engineering at the University of Stellenbosch

Applied Mathematics, Department of Mathematics Sciences
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Dr K.M. Hunter

March 2008

Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature: . . . .
W. Singels

Date: . . . .

Abstract

When building or improving a petrochemical plant, drawings are used extensively in the design process. However, existing petrochemical plants seldom match their drawings, or the drawings are lost, forcing the need to generate a 3D model of the structure of the plant. In this thesis photogrammetry is investigated as a method of generating a digital 3D model of an existing plant. Camera modeling, target extraction and 3D reconstruction are discussed in detail, and a real-world system is investigated.

Opsomming

Met die bou of uitbreiding van 'n bestaande petro-chemiese aanleg word 'n 3D model van die aanleg benodig vir die ontwerp proses. Dit gebeur egter dikwels dat die planne en tekeninge van die aanleg nie ooreenstem met die fisiese aanleg nie, of soms net verlore is. In hierdie tesis word fotogrametrie ondersoek as 'n tegniek om 'n digitale 3D model van bestaande aanlegte te skep. Kamera modelering, teiken ekstraksie en 3D rekonstruksie word verder bespreek en 'n werklike sisteem word ondersoek.

Acknowledgements

The author would like to thank the following people for their contribution toward this project.

- Dr. Karin Hunter for her guidance throughout this process
- Prof. Ben Herbst for his wisdom
- My family for their continued support
- My housemates in 2007 for distracting me when necessary
- Bhukka (Pty) Ltd for financial support and technical input

Contents

Abbreviations

- AC - Absolute Conic
- CAD - Computer Aided Design
- CCD - Charge-Coupled-Device
- DDE - Dynamic Data Exchange
- DLT - Direct Linear Transformation
- IAC - Image of the Absolute Conic
- RMS - Root-Mean-Squared
- SVD - Singular Value Decomposition

Chapter 1

Introduction

1.1 Background

The problem we are examining comes from the petrochemical industry. All petrochemical plants, see Figure 1.1 for example, are built from drawings, designs and plans. Often the final product does not match the drawings in every detail. Also, as the science in the industry improves, older plants need to be updated, but the accompanying drawings are not updated accordingly: the real world status of the plant does not correlate with the drawings. In many cases the original drawings are simply missing or lost.

The problem is thus to create an accurate description of the plant's as-is status. A manual survey of the complete site is difficult, expensive, would take too long and is not guaranteed to be accurate. A better and cheaper alternative is to model the plant digitally. A digital model has the added advantage that modifications to the plant can be planned on computers using Computer Aided Design (CAD), as is standard in the industry these days.

This thesis examines the application of a photogrammetry system in this environment.

1.2 Literature Review

Back in the sixties it was thought that making computers see would be a simple problem to solve and would take a year or less to complete. The opposite has proven to be true as, to date, no-one has been able to digitally mimic the complexity of human vision process in full. However, recent progress in the

Figure 1.1: An example of a petrochemical plant

field of Computer Vision has allowed researchers to obtain some outstanding successes reconstructing 3D environments from 2D images. The following is a look at the biggest recent steps forward in each field within Computer Vision.

1.2.1 Camera Calibration

Camera Calibration is a necessary, and usually the first, step to extract metric information from a 2D image. The aim of calibration is to numerically describe the mapping of a 3D environment to a 2D image. This computation of the camera projection matrix is also known as *resectioning*.

Calibration techniques can be broadly classified as either linear [10, 9] or non-linear [22]. Non-linear algorithms have the disadvantage of being computationally expensive and requiring good initial estimates. With a bad initial estimate the algorithm will not converge or may become trapped in a local minimum. Linear calibration techniques, although not as accurate, are more intuitive, faster and do not require initial estimates, therefore we prefer them above non-linear methods. Linear methods of calibration can be divided into two further categories - Self-calibration and Object-calibration.

Object-calibration The camera is calibrated using a calibration object of which the 3D metric information is available to high precision. The calibration object is usually two or three orthogonal planes and sometimes even a cube. Calibration from such an object is very accurate and can be done from only a single image [8]. Some algorithms use a single plane and a precise, known rotation of the plane [31]. Manufacturing of such calibration objects is expensive because of the required accuracy levels and the calibration process can soon become too elaborate for a real-world application.

Self-calibration Calibration in this category is done without the use of a calibration object. The camera is moved through a static environment and the rigidity of the scene provides enough constraints [21] to derive some of the camera parameters. If the internal parameters of the camera are the same in all the images, one can find both internal and external camera parameters from just three images, providing there are enough corresponding points in the images [19]. This approach, although very flexible, can lead to large errors that are hard to interpret [3].

Zhang calibration The algorithm we have chosen to implement is somewhere between Object- and Self-Calibration. Using a planar calibration grid shown in a few (at least three) different views and homographies (see Section 3.1) we estimate the internal parameters of the camera. This technique was first proposed by Liebowitz and Zisserman in 1998 [16]. Zhang proposed an improved method [39] in 2000 and this is the one implemented here. There are other similar algorithms [30] and although they are more flexible in many ways, they have some difficulties with initialisation.

Although not implemented here, the reader might want to consider the effect of lens distortion on camera calibration. Based on certain papers in the literature [5, 31, 32], it is likely that the distortion function is totally dominated by the radial components. It has also been found that the improvement from modeling additional distortion is negligible when compared with sensor quantization and can also cause numerical instability [31, 32].

1.2.2 Fundamental matrix

The fundamental matrix describes the relationship between two cameras and is the basis on which stereo vision is built. The essential matrix is a special form of the fundamental matrix where the internal parameters of both cameras are known. It was first presented in 1981 by Longuet-Higgins [17] before the fundamental matrix was known. The realisation that the essential matrix could be applied in the uncalibrated case as well, giving the fundamental matrix, was published simultaneously by Faugeras [7] and Hartley [11, 13] in the early 1990s. A good summary of the most common uncalibrated methods is given by Zhang in his review of epipolar geometry [38]. The detailed study of the basic 8-point normalised algorithm, which is also implemented here, is given by Hartley in [12].

Apart from the articles mentioned above, special mention has to made of the book *Multiple View Geometry in Computer Vision* by Hartley and Zisserman [15]. This book provided an easy reference guide and an exhaustive coverage of the results obtained by them and other researchers worldwide on all relevant areas. As far as possible we follow the notation used by Hartley and Zisserman in this book in our description of algorithms and in our equations.

1.2.3 Image Processing

Image processing algorithms are used to extract targets from images using techniques like edge detection, contour detection, thresholding, dilation and erosion. The Open Source Computer Vision Library package[1] (OpenCV) created by Intel was used extensively for this.

1.3 Overview

Making a complete 3D model of a real world scene requires a complicated system with many stages. The basic outline of such a system is shown in Table 1.1.

Camera Calibration
Data Acquisition
Target Extraction
Find Stereo Pairs
Feature Extraction
Reconstruction

Table 1.1: Stages in 3D Reconstruction

[1]This package is available from http://www.sourceforge.net/projects/opencvlibrary

We describe in detail different camera calibration techniques in Chapter 3 and select the most appropriate one for this system. Different methods of reconstruction using stereo vision are investigated in Chapter 4 and the results of a simple implementation are given at the end of the chapter. In Chapter 5 we examine the system used by Bhukka (Pty) Ltd. We improve the efficiency of this system by constructing a robust and accurate algorithm to automatically extract coded targets from images. The results obtained from this algorithm are given in the last section of the chapter. Finally we draw some conclusions in Chapter 6.

Chapter 2

System Overview

The first steps toward a practical system for 3D reconstruction is to find the appropriate tools to capture real world data. Thereafter software can be used to process the gathered data into a complete 3D model.

2.1 Hardware

Traditionally, reconstruction systems were built around specialised hardware such as laser range finders or stereo camera rigs with customised cameras. Recent progress in consumer digital hardware has allowed a cheaper alternative. The use of off-the-shelf digital photo- or video cameras along with increasingly powerful software allows the accuracy of such systems to near those of their expensive counterparts. We now discuss which one is best for this specific application.

2.1.1 Laser Range Scanner

A Laser Range scanner system usually consists of a single laser range finder, see Figure 2.1, attached to a portable computer. The scanner scans a set area collecting millions of data points that are fed into the computer. Because a typical scanner can scan 360 degrees in the horizontal plane and about 300 degrees in the vertical plane it can be placed in a spot and then scan the surrounding area. To gather a complete 3D model of a plant the scanner is placed in a number of positions surrounding the area of interest. This is a time consuming process as a scan of a small 7×7 meter area can take up to two hours [29]. Laser range scanners make very accurate measurements but because of the many overlapping areas, the gathered data contains a lot of redundant information. Laser scanners are also very expensive and are rather cumbersome to get into hard-to-reach places because of their size. Due to its size, holes and occlusion also frequently occur in the data, which makes the data gathering process a tedious task.

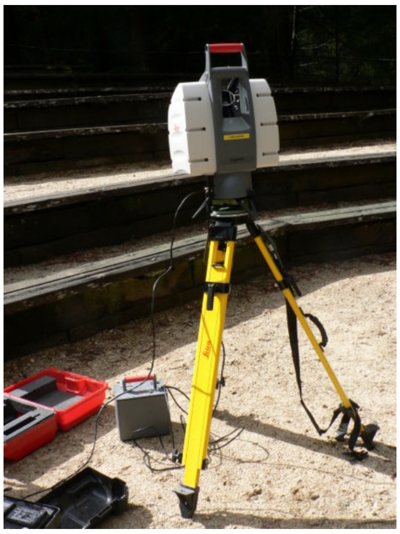


Figure 2.1: An example of a Laser Range scanner [34]

2.1.2 Digital Camera

Digital camera technology has become cheap and sufficiently accurate in the last few years. The camera, see Figure 2.2, gives easy access to places the laser scanner cannot be used. We can either have a single camera or a stereo camera rig. A stereo rig would consist of two cameras placed in fixed positions relative to each other. The slightest change in one camera's position relative to the other will render the system paralyzed until it is recalibrated. It can also become a bulky system, which once again, can not get into small spaces or higher places as easy as a single camera. Such a sensitive system is therefore not practical for our setup.

With a single camera the capture process is simpler. The camera is calibrated to determine its internal parameters and then photos are taken of the entire plant. The relative positions of the images to each other can be calculated by software for each separate case, after which objects can be selected and reconstructed out of the image pairs.

Figure 2.2: An example of a digital camera [36]

2.1.3 Digital Video Camera

If we view a digital video simply as a series of photos then reconstruction from the video camera, see Figure 2.3, works on the same principal as the still digital camera. The only difference being the abundance of the data. More data should mean increased accuracy, but in this case the accuracy gained is outweighed heavily by the inefficiency of sorting through the data. A single image taken from such a series will have a low resolution as video streams do not save all the information in each frame. There are techniques to solve this problem [28], but they are overly complicated and not robust enough for application here.

Property	Laser Range Finder	Digital Camera	Video Camera
Price	Highly Expensive	Cheap	Cheap
Mobility	Impractical	Very	Very
Accuracy	Very High	Sufficient	Average
Data	Cumbersome	Specific	Cumbersome

Table 2.1: Choices for Physical System

From Table 2.1 we conclude that a single digital camera with a high resolution is the best choice for this application. Note that sufficient accuracy is only guaranteed if surveyed targets are used in the photos (see Chapter 5).

Figure 2.3: An example of a digital video camera [35]

2.2 Software

Once the data is collected it needs to be processed by software before it is of any value. There are several steps in the system that require software - camera calibration, target extraction and the reconstruction process. There are a multitude of algorithms for each step. Here we briefly discuss the techniques we cover in more detail in later chapters.

All of the software written for this thesis was coded in Python 2.5 [1] and is on the attached CD under the *PythonCode* directory.

2.2.1 Camera Calibration: Linear Calibration

Linear calibration algorithms estimate the camera projection matrix from image coordinates and their corresponding known 3D points. Similarly, the camera projection matrix may also be determined from known world lines and corresponding lines in the images. These algorithms are only valid under the assumption that there is no lens distortion. The topic of modeling radial distortion will be covered later. Only one image is needed to do calibration, but the image needs to contain a clear view of a calibration object and the points on the calibration object cannot be co-planar.

2.2.2 Camera Calibration: Planar Calibration

Planar calibration uses multiple images of a planar object (calibration grid) and homographies to estimate the internal camera parameters. The size of

[1] Python language is available from http://www.python.org/download

the calibration grid needs to be known accurately. The algorithm only provides the internal parameters of the camera, but for our setup this is sufficient as the external parameters change for each image anyway. An implementation and detailed explanation is given in Section 3.4.

2.2.3 Target Extraction

There are two main approaches to extract targets from images - image processing and pattern recognition. Image processing involves different filtering and shape detection algorithms to find the location of the targets in an image and then identifying the targets. Pattern Recognition algorithms start with a specific target and try to locate that target in each image. We implement a automatic extraction algorithm using image processing techniques in Chapter 5.

2.2.4 Reconstruction

There are many algorithms for reconstruction depending on what information you start with. There are also reconstruction algorithms that use more than two images (trifocal tensor) but we focus solely on two view geometry (stereo vision). The fundamental matrix describes the relationship between the two cameras. As is evident from the literature study there are many different methods of determining the fundamental matrix. We discuss the most relevant techniques and code a simple implementation.

2.3 Conclusions

In this chapter we considered both the hardware and software requirements for a practical photogrammetry system in the petrochemical industry. Due to the inherent convoluted nature of petrochemical plants we decided that a single camera is the most mobile unit for our hardware: both the stereo camera rig and the laser range finder being unwieldy and bulky and the video camera being overly complex. To increase accuracy we used surveyed targets in the images, see Chapter 5 for our motives and a detailed explanation of the target extraction algorithm.

In terms of software, we implemented Zhang's planar calibration algorithm to find the internal camera parameters. The internal parameters are then used along with the normalised 8-point algorithm to do 3D reconstruction.

Chapter 3

Camera Model and Calibration

The basic idea of camera calibration is to mathematically model the process by which a digital camera creates a 2D image of a 3D scene. The camera projection matrix P is a 3x4 matrix so that

$$\boldsymbol{x} = P\boldsymbol{X}$$

where $\boldsymbol{X}$ is a homogeneous 4-vector $(X, Y, Z, 1)^T$ representing a real world coordinate and $\boldsymbol{x}$ is a homogeneous 3-vector $(x, y, 1)^T$ which is the pixel in the image. For an explanation of homogeneous coordinates, see Appendix A.2. The camera matrix is divided into two parts - the internal parameters K and the external parameters of rotation R and camera center $\widetilde{\boldsymbol{C}}$.

The problem of calibration has many similarities to the computation of homographies. We therefore present the theory of homographies first before moving on to linear calibration.

3.1 Homographies

In essence the calculation of an homography is an estimation problem. The process of estimation in this context meaning that we compute a transformation or mathematical quantity based on some measurements. The simplest form of an homography is a 2D to 2D homography given by Hartley and Zisserman [15, Chapter 4] as:

Given a set of points $\boldsymbol{x}_i$ in $\mathbb{P}^2$ and a corresponding set of points $\boldsymbol{x}'_i$ likewise in $\mathbb{P}^2$, compute the projective transformation that takes each $\boldsymbol{x}_i$ to $\boldsymbol{x}'_i$.

For an explanation of projective space $\mathbb{P}^n$ see Appendix A.1. In the 2D to 2D case the projective transformation is a 3×3 matrix H such that $H\boldsymbol{x}_i = \boldsymbol{x}'_i$ for each i. Each point correspondence provides two constraints on H. The matrix H has 9 entries, but is defined only up to scale, which gives us eight degrees of freedom. Therefore, if four correspondences are given then an exact solution can be found.

Direct Linear Transformation (DLT) Algorithm

The DLT is the simplest linear algorithm for computing H from four 2D to 2D homogeneous point correspondences. We can express the equation $H\boldsymbol{x}_i = \boldsymbol{x}'_i$ in terms of vector cross product as $\boldsymbol{x}'_i \times H\boldsymbol{x}_i = \boldsymbol{0}$.

If each row in H is given as $\boldsymbol{h}^{jT}$, with $j = 1, 2, 3$ in this case, then we may write

$$H\boldsymbol{x}_i = \begin{pmatrix} \boldsymbol{h}^{1T}\boldsymbol{x}_i \\ \boldsymbol{h}^{2T}\boldsymbol{x}_i \\ \boldsymbol{h}^{3T}\boldsymbol{x}_i \end{pmatrix}.$$

Expanding the vector $\boldsymbol{x}'_i$ to $(x'_i, y'_i, w'_i)^T$, the cross product is given as

$$\boldsymbol{x}'_i \times H\boldsymbol{x}_i = \begin{pmatrix} y'_i\boldsymbol{h}^{3T}\boldsymbol{x}_i - w'_i\boldsymbol{h}^{2T}\boldsymbol{x}_i \\ w'_i\boldsymbol{h}^{1T}\boldsymbol{x}_i - x'_i\boldsymbol{h}^{3T}\boldsymbol{x}_i \\ x'_i\boldsymbol{h}^{2T}\boldsymbol{x}_i - y'_i\boldsymbol{h}^{1T}\boldsymbol{x}_i \end{pmatrix}.$$

Since $\boldsymbol{x}'_i \times H\boldsymbol{x}_i = \boldsymbol{0}$ and we can write $\boldsymbol{h}^{jT}\boldsymbol{x}_i = \boldsymbol{x}_i^T\boldsymbol{h}^j$ for $j = 1, 2, 3$, we have three equations in the unknown $\boldsymbol{h}$, which may be written as

$$\begin{bmatrix} \boldsymbol{0}^T & -w'_i\boldsymbol{x}_i^T & y'_i\boldsymbol{x}_i^T \\ w'_i\boldsymbol{x}_i^T & \boldsymbol{0}^T & -x'_i\boldsymbol{x}_i^T \\ -y'_i\boldsymbol{x}_i^T & x'_i\boldsymbol{x}_i^T & \boldsymbol{0}^T \end{bmatrix} \begin{pmatrix} \boldsymbol{h}^1 \\ \boldsymbol{h}^2 \\ \boldsymbol{h}^3 \end{pmatrix} = \boldsymbol{0} \tag{3.1.1}$$

where $\boldsymbol{h}$ is a 9-vector made up of the entries of matrix H,

$$\boldsymbol{h} = \begin{pmatrix} \boldsymbol{h}^1 \\ \boldsymbol{h}^2 \\ \boldsymbol{h}^3 \end{pmatrix}, \qquad H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{bmatrix} \tag{3.1.2}$$

with h_i the i-th element of $\boldsymbol{h}$.

Note that although there are three equations in (3.1.1) only two of them are linearly independent (the third row is, up to scale, the sum of the first two). This confirms that each point correspondence only gives two constraints on the vector $\boldsymbol{h}$. The set of equations in (3.1.1) now becomes

$$\begin{bmatrix} \mathbf{0}^T & -w'_i\boldsymbol{x}_i^T & y'_i\boldsymbol{x}_i^T \\ w'_i\boldsymbol{x}_i^T & \mathbf{0}^T & -x'_i\boldsymbol{x}_i^T \end{bmatrix} \begin{pmatrix} \boldsymbol{h}^1 \\ \boldsymbol{h}^2 \\ \boldsymbol{h}^3 \end{pmatrix} = \mathbf{0} \tag{3.1.3}$$

which, for brevity's sake, will be given as $A_i\boldsymbol{h} = \mathbf{0}$ where A_i is the 2×9 matrix of (3.1.3).

If we have the minimum of four corresponding points we obtain the equation $A\boldsymbol{h} = \mathbf{0}$ by stacking each A_i to form A. Whether we use (3.1.1) or (3.1.3) the matrix A has rank 8, and thus has a 1-dimensional null-space which provides a non-zero solution for $\boldsymbol{h}$. The solution can only be determined up to a non-zero scale factor, but H is normally only determined up to scale. So the solution $\boldsymbol{h}$ gives the required H with a scale for the vector chosen by some requirement on its norm such as $\|\boldsymbol{h}\| = 1$.

In a practical situation the measurements of the points in the images will not be exact due to noise. If more than four point correspondences are given, with the presence of noise, we have an over-determined system and the solution will not be exact anymore. By minimising some cost function we try to find the 'best' possible approximation for the vector $\boldsymbol{h}$. In Appendix B it is shown that this solution is the unit singular vector corresponding to the smallest singular value of A.

Normalising transformations

In this section we describe a method of normalising data by means of translation and rotation of the image coordinates. The normalisation, called root-mean-squared (RMS) normalisation, is carried out before the DLT algorithm, and subsequently the resulting H is denormalised to return it to the original coordinate system. The reason normalisation is important to the accuracy of the results is that the result of the DLT algorithm depends on the coordinate frame in which the points are expressed. It was shown by Hartley that the result is not invariant under similarity transformations of the image [12]. In other words, some coordinate systems will deliver more accurate results for the homography.

In the first step of normalisation the coordinates are translated (by a different translations for each image) to map the centroid of all the points to the origin. The second step is to scale the points so they have the form $\boldsymbol{x} = (x, y, w)^T$, with x,y and w having the same average magnitude. This is called **isotropic scaling**, derived from the Greek *iso* (equal) and *tropos* (turn, way). The usual distance to which 2D points are scaled is $\sqrt{2}$ from the origin. This means the 'average' homogeneous point is equal to $(1, 1, 1)^T$. Now the transformation is applied to each image independently.

The complete DLT algorithm, including normalisation, is summarised in Figure 3.1 and the code is given in the *ThreeDVision.Homography* function. Note that the function is generalised to handle N-dimensional to N-dimensional homographies.

3.2 Linear Calibration

The equations for finding the camera projection matrix are very similar to those for computing a homography. The biggest difference being that instead of a 2D to 2D mapping, we now deal with a 3D to 2D mapping. We start once again with point correspondences, but this time $\boldsymbol{X}_i \leftrightarrow \boldsymbol{x}_i$ where $\boldsymbol{X}_i$ is a 4-vector homogeneous point in 3D space and $\boldsymbol{x}_i$ is a 3-vector homogeneous 2D point on the image. The matrix P is thus 3×4 such that $\boldsymbol{x}_i = P\boldsymbol{X}_i$ for all i.

As in the previous section for each correspondence $\boldsymbol{X}_i \leftrightarrow \boldsymbol{x}_i$ we derive the relationship

$$\begin{bmatrix} \mathbf{0}^T & -w_i\boldsymbol{X}_i^T & y_i\boldsymbol{X}_i^T \\ w_i\boldsymbol{X}_i^T & \mathbf{0}^T & -x_i\boldsymbol{X}_i^T \\ -y_i\boldsymbol{X}_i^T & x_i\boldsymbol{X}_i^T & \mathbf{0}^T \end{bmatrix} \begin{pmatrix} \boldsymbol{P}^1 \\ \boldsymbol{P}^2 \\ \boldsymbol{P}^3 \end{pmatrix} = \mathbf{0} \qquad (3.2.1)$$

where each P^i is now a 4-vector. The equations in (3.2.1) are also linearly dependent, as for homographies, so once again we use only the first two:

$$\begin{bmatrix} \mathbf{0}^T & -w_i'\boldsymbol{x}_i^T & y_i'\boldsymbol{x}_i^T \\ w_i'\boldsymbol{x}_i^T & \mathbf{0}^T & -x_i'\boldsymbol{x}_i^T \end{bmatrix} \begin{pmatrix} \boldsymbol{h}^1 \\ \boldsymbol{h}^2 \\ \boldsymbol{h}^3 \end{pmatrix} = \mathbf{0}. \qquad (3.2.2)$$

Since the matrix P has 12 entries and is only defined up to an unknown

Objective

Given $n \geq 4$ 2D to 2D point correspondences $\{\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}'_i\}$, determine the 2D homography matrix such that $H\boldsymbol{x}_i = \boldsymbol{x}'_i$.

Algorithm

1. **Normalisation of $\boldsymbol{x}$**: Compute a similarity transformation T, consisting of a translation and scaling, that takes $\boldsymbol{x}$ to a new set $\tilde{\boldsymbol{x}}$ such that the centroid is at the origin and the average distance from the origin is $\sqrt{2}$.
2. **Normalisation of $\boldsymbol{x}'$**: Compute a similar transformation T' to take $\boldsymbol{x}'$ to a new set of points $\tilde{\boldsymbol{x}}'$.
3. **Compute A_i**: For each correspondence $\tilde{\boldsymbol{x}}_i \leftrightarrow \tilde{\boldsymbol{x}}'_i$ compute the matrix A_i from (3.1.3).
4. Assemble the n 2×9 matrices A_i into a single $2n \times 9$ matrix A.
5. **Solve for $\boldsymbol{h}$**: Calculate the SVD of A. The unit singular vector corresponding to the smallest singular value is the solution $\boldsymbol{h}$ (see Appendix B).
6. **Find $\widetilde{H}$**: The matrix $\widetilde{H}$ is determined from $\boldsymbol{h}$ as in (3.1.2).
7. **Denormalisation**: Set $H = T'^{-1}\widetilde{H}T$

Figure 3.1: DLT Algorithm including RMS normalisation.

scale, P has 11 degrees of freedom. Each correspondence gives two constraints, so we require at least 6 points if no further knowledge of the camera parameters is available. Again, the point sets are normalised so their centroids are at the origin and the root-mean-squared (RMS) distance from the origin is scaled to $\sqrt{2}$, in the case of $\boldsymbol{x}_i$, and $\sqrt{3}$, for $\boldsymbol{X}_i$.

Using the equation (3.2.2) and the RMS normalisation, the algorithm for computation of the camera matrix P proceeds in the same manner as that for H. The code used for linear calibration is presented in *ThreeDVision.LinearCalibration*.

A slight gain in accuracy can be made if an iterative minimisation method, particularly the Levenberg-Marquardt technique [23], is used to minimise the geometric error. Seeing as we are interested in a simple implementation and the normalised DLT algorithm delivers sufficiently accurate

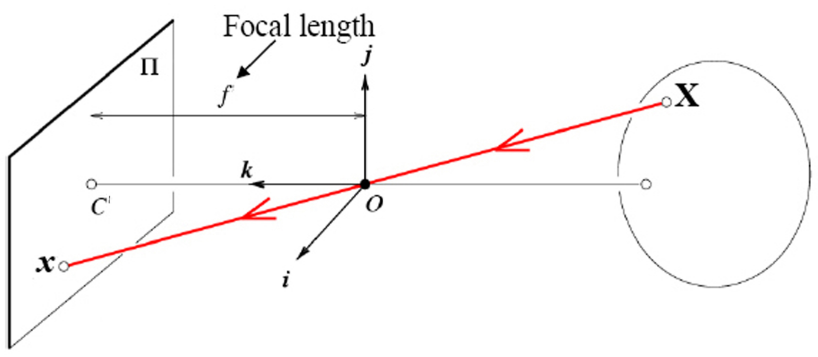


Figure 3.2: Pinhole Camera Geometry

results, such techniques are deemed outside the scope of this thesis.

3.3 Camera Model

Now that we have covered the procedure of computing the camera projection matrix (P), we spend some time discussing what information is contained within the matrix. What do we know about the camera when we have P? To start off, we describe the simplest camera model, the basic pinhole model. We generalise this to model Charged-Couple-Device (CCD) cameras and finally we show how to decompose the matrix P into camera orientation and the internal parameters.

3.3.1 Basic Pinhole Camera

The pinhole camera has no lens to focus light and so the camera aperture is described as a single point. The process is illustrated in Figure 3.2. We place the origin of an Euclidean coordinate system at the center of projection O. Consider the plane $\Pi = f$, which is called the *image plane* or *focal plane*. The projection of a point, $\boldsymbol{X} = (X, Y, Z)^T$, will be at the point $\boldsymbol{x}$ where a line joining $\boldsymbol{X}$ to the center of projection O meets the image plane Π. The center of projection, O in this case, is called the *camera center* or *optical center*. The line from the camera center perpendicular to the image plane is called the *optical axis* and meets the image plane at the *principal point* C'.

The point $(X,Y,Z)^T$ is mapped to the point $(fX/Z, fY/Z, 1)^T$ on the image plane. If we write the vectors in homogeneous coordinates we can express the central projection mapping in terms of matrix multiplication

$$\begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & & & 0 \\ & f & & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3.3.1}$$

Up until now we assumed the origin of the coordinates in the image plane is at the principal point. To generalise this, we add a term to the mapping, which becomes

$$(X,Y,Z)^T \mapsto \left(fX/Z + p_x, fY/Z + p_y\right)^T$$

where $(p_x, p_y)^T$ are the coordinates of the principal point. Expressing this equation in homogeneous coordinates

$$\begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f & & p_x & 0 \\ & f & p_y & 0 \\ & & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}. \tag{3.3.2}$$

If we write

$$K = \begin{bmatrix} f & & p_x \\ & f & p_y \\ & & 1 \end{bmatrix}$$

then (3.3.2) has the concise form

$$\boldsymbol{x} = K\,[\, I \mid \mathbf{0} \,]\, \boldsymbol{X}_{cam}. \tag{3.3.3}$$

The matrix K is called the *camera calibration matrix*. We have written $(X,Y,Z,1)^T$ as $\boldsymbol{X}_{cam}$ in (3.3.3) to emphasize that the camera is assumed to be at the origin with the principal axis pointing straight down the Z-axis, and that the point $\boldsymbol{X}_{cam}$ is expressed in this coordinate system. This coordinate system is called the *camera coordinate frame*.

In general, points in 3D space will be expressed in terms of a *world coordinate frame*. The camera coordinate frame and the world coordinate frame are related by a simple translation and rotation. If we let R represent a 3×3 rotation matrix and $\widetilde{\boldsymbol{C}}$ the coordinates of the camera center in the world coordinate frame, then formula (3.3.3) becomes

$$\boldsymbol{x} = KR\left[I| - \widetilde{\boldsymbol{C}}\right]\boldsymbol{X}. \tag{3.3.4}$$

The parameters contained in K are called the *internal parameters* of the camera and the parameters that describe the camera position in the world coordinate frame, R and $\widetilde{\boldsymbol{C}}$, are the *external parameters*.

3.3.2 CCD Cameras

The pinhole camera model is described by (3.3.4), but the model assumes that the image coordinates have equal scales in both axial directions. In the case of a charge-coupled-device (CCD) camera, this is often not the case. There is a possibility that the pixels of a digital camera are non-square. If the number of pixels per unit distance in image coordinates are m_x and m_y in the x and y directions, then the transformation from world to pixel coordinates is simply a multiplication by an extra factor $\operatorname{diag}(m_x, m_y, 1)$. We also add the parameter s that is referred to as the *skew* parameter. The skew parameter is almost always zero, except in rare cases such as an image taken of another image.

The general form of the calibration matrix K for CCD cameras now becomes

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ & \alpha_y & y_0 \\ & & 1 \end{bmatrix} \tag{3.3.5}$$

where $\alpha_x = fm_x$ and $\alpha_y = fm_y$ represents the focal length of the camera in terms of pixels. Similarly, x_0 and y_0 represent the principal point in terms of pixel dimensions. A finite projective camera for which the calibration matrix K is of the form (3.3.5) can now be written as

$$P = KR\left[I| - \widetilde{\boldsymbol{C}}\right]. \tag{3.3.6}$$

3.3.3 Decomposition of the Camera Matrix

If the camera matrix is calculated from (3.3.6) then the parameters are immediately available and decomposition is clearly unnecessary. However, there are cases when a camera matrix is calculated without such information readily available, for example direct calibration from world coordinates. In such cases we wish to find the external parameters (camera center and orientation) as well as the internal parameters from just the matrix P.

First we recover the camera center, $\widetilde{\mathbf{C}}$. For the camera center the equation $P\widetilde{\mathbf{C}} = 0$ holds since $\widetilde{\mathbf{C}}$ is the only point that can not be imaged by the camera. This equation may be solved by finding the right null-vector of the matrix P by means of SVD (see Appendix B).

To find the orientation and internal parameters we first write equation (3.3.6) as

$$P = KR[I| - \widetilde{\mathbf{C}}] = K[R| - R\widetilde{\mathbf{C}}] = [M| - M\widetilde{\mathbf{C}}].$$

We decompose the matrix M into K and R by using RQ-Decomposition. This decomposition gives us an upper-triangular matrix which represents K and a orthogonal matrix which is R. After doing the RQ-Decomposition it is crucial to remove the ambiguity by assuring that K has positive diagonal entries. Matrix K then has the form of equation (3.3.5).

The code for this function is given in *ThreeDVision.CamMatrixDecomp* on the attatched CD.

3.4 Planar Calibration

In certain cases it may not be possible to perform a full linear calibration. If the calibration object is not available or there are time constraints, a planar calibration algorithm can provide a quick method of performing a limited calibration. If a single camera is used and the focal length of the camera is not changed between images, the internal parameters of the camera will be constant. Planar calibration provides us with an algorithm to determine only the internal parameters.

The Image of the Absolute Conic

The absolute conic (AC), see Appendix A.3, and its projection in images (IAC), see Figure 3.3, are important concepts in the process of planar calibration. Since the AC is invariant under Euclidean transformations, its relative position to a moving camera is constant. For constant internal parameters its image will therefore also be constant. It can be seen as a calibration object which is naturally present in all scenes.

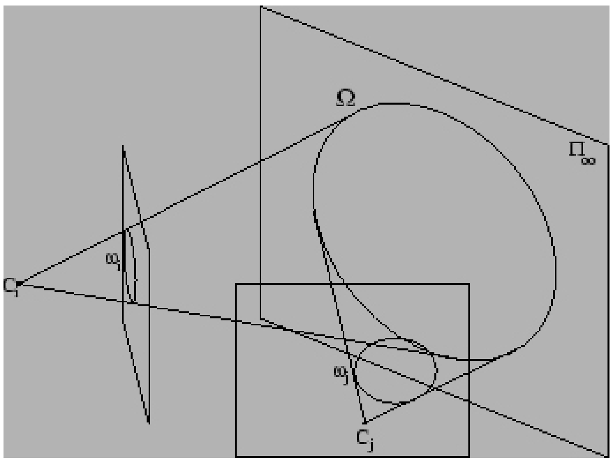


Figure 3.3: The Absolute Conic (Ω) on the plane at infinity (Π_∞) and its projection in the images (ω_i and ω_j)

We can derive a relationship between the calibration matrix K and the IAC, $\boldsymbol{\omega}$. First we determine the map between the plane at infinity, Π_∞, and the camera image plane. Points on Π_∞ may be written as $\boldsymbol{X}_\infty = \left(\boldsymbol{d}^T, 0\right)^T$, and are imaged by a general camera (3.3.6) as

$$\boldsymbol{x} = P\boldsymbol{X}_\infty = KR[I| - \widetilde{\boldsymbol{C}}] \begin{pmatrix} \boldsymbol{d} \\ 0 \end{pmatrix} = KR\boldsymbol{d}.$$

This shows that the mapping between Π_∞ and an image is given by the planar homography $\boldsymbol{x} = H\boldsymbol{d}$ with $H = KR$.

The IAC is also unaffected by the location and orientation of the camera. If T_i is a projective transformation from one image $J - 0$ to image J_i, then in particular it must take a point on the IAC in one image to a point on the IAC in the other. In short, T_i must preserve the IAC.

We also know that under a point homography, $\boldsymbol{x} \mapsto H\boldsymbol{x}$, a conic C maps as $C \mapsto H^{-T}CH^{-1}$. It follows that the AC which is a conic $C = \Omega_\infty = I$ and is the identity matrix on the plane Π_∞, maps to

$$\boldsymbol{\omega} = (KR)^{-T} I \, (KR)^{-1} = K^{-T}RR^{-1}K^{-1} = \left(KK^T\right)^{-1}.$$

In other words the IAC is $\boldsymbol{\omega} = \left(KK^T\right)^{-1}$ in $J - i$ for all i.

Another way of considering the AC is by geometric interpretation of the rigidity constraints. If we consider the epipolar planes in Figure 3.4, given by

Π_1 and Π_2, which are both tangent to $C = \Omega_\infty = I$, then the corresponding epipolar lines l'_1 and l'_2 are tangent to the IAC, ω', in the second image. This is due to the epipolar constraint, contained in the fundamental matrix F (see Section 4.1)

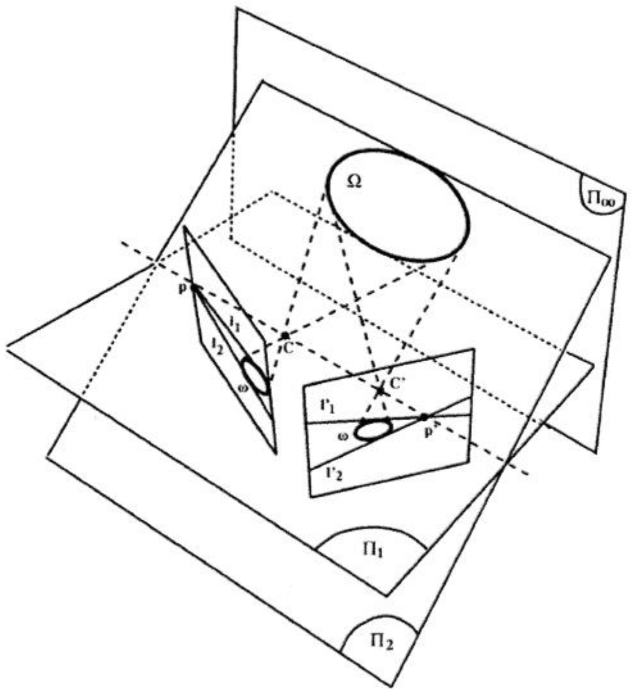


Figure 3.4: The absolute conic and epipolar geometry

Computing K

The image of at least three squares (or three images of a square) provides sufficiently many constraints to compute K. We can also use images of a calibration plane with known dimension, such as those in Figure 3.10. We know that the sides of the square or calibration plane are parallel in the real world. This gives us the information needed to define the images of the plane at infinity. The correspondences between its four corners and their imaged corners define a homography H between the plane at infinity and the image. Applying this homography to circular points $(1, \pm i, 0)^T$ on π determines their images as $H(1, \pm i, 0)^T$. We now have two constraints on the yet unknown ω. We need five constraints to determine a conic, therefore the requirement of three homographies. Once the conic ω is determined we can compute K using Cholesky factorisation (for a proof, see [15, Section A4.2.1]).

The algorithm for planar calibration is summarised in Figure 3.5 and the code to perform calibration given the homographies is given in *ThreeDVision.PlanarCalibration*.

Objective

Given at least 3 images of a square, compute the internal parameters K.

Algorithm

1. Represent ω as a homogeneous 6-vector $\boldsymbol{b} = (B_1, B_2, B_3, B_4, B_5, B_6)^T$ where:
$$\omega = \begin{bmatrix} B_1 & B_2 & B_4 \\ B_2 & B_3 & B_5 \\ B_4 & B_5 & B_6 \end{bmatrix}$$
2. Write each homography as $H = [\boldsymbol{h}_1, \boldsymbol{h}_2, \boldsymbol{h}_3]$. The two constraints on ω are then $\boldsymbol{h}_1^T \omega \boldsymbol{h}_2 = 0$ and $\boldsymbol{h}_1^T \omega \boldsymbol{h}_1 - \boldsymbol{h}_2^T \omega \boldsymbol{h}_2 = 0$. Each of these constraints may be written as $\boldsymbol{a}^T \boldsymbol{b} = 0$. For example, $\boldsymbol{u}^T \omega \boldsymbol{v} = 0$, where $\boldsymbol{u} = (u_1, u_2, u_3)^T$ and $\boldsymbol{v} = (v_1, v_2, v_3)^T$, the 6-vector $\boldsymbol{a}$ is given by
$$\boldsymbol{a} = (v_1 u_1, v_1 u_2 + v_2 u_1, v_2 u_2, v_1 u_3 + v_3 u_1, v_2 u_3 + v_3 u_2, v_3 u_3)^T .$$
3. Stack equations $\boldsymbol{a}^T \boldsymbol{b} = 0$ from each constraint in the form $A\boldsymbol{b} = 0$, where A is a $n \times 6$ matrix for n constraints.
4. Solve for $\boldsymbol{b}$ using the SVD (see Appendix B). This gives ω.
5. Decompose ω into K using matrix inversion and Cholesky factorisation.

Figure 3.5: Planar Calibration Algorithm.

3.5 Implementation

3.5.1 Linear Calibration

We calibrate the camera using the *ThreeDVision.LinearCalibration* function, an implementation of the algorithm given in Section 3.2, along with the calibration object constructed of Lego blocks. We know the size of each block is

$32 \times 16 \times 9.6$mm and so one brick unit in the x and y directions is 32mm, one unit in the z direction is 9.6mm. The origin of the world coordinate system is the front lower corner of the object with the x-axis running to the right, the y-axis to the left, and the z-axis is vertical, as in Figure 3.6.

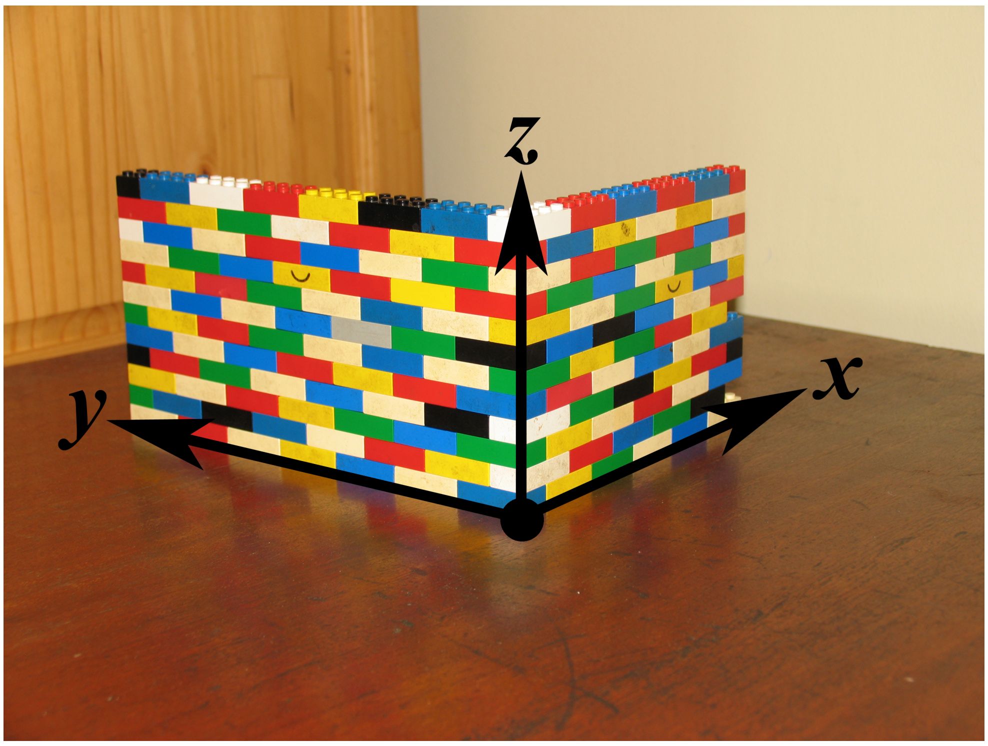


Figure 3.6: Calibration object with coordinate system

Although a minimum of only 6 features are required to calculate the camera projection matrix P, we use as many as possible over as much of the frame as possible to get an accurate result. A rule of thumb is that for a good estimation the number of constraints, or points, should exceed the number of unknowns by a factor of five. This means at least 28 points are required for a good estimate. The projection matrix computed from all the points in Figure 3.7 is

$$P_1 = \begin{bmatrix} 6.195 & -4.745 & -0.118 & 1620.550 \\ 0.556 & -0.117 & -7.535 & 1708.798 \\ 0.001 & 0 & 0.002 & 1 \end{bmatrix}.$$

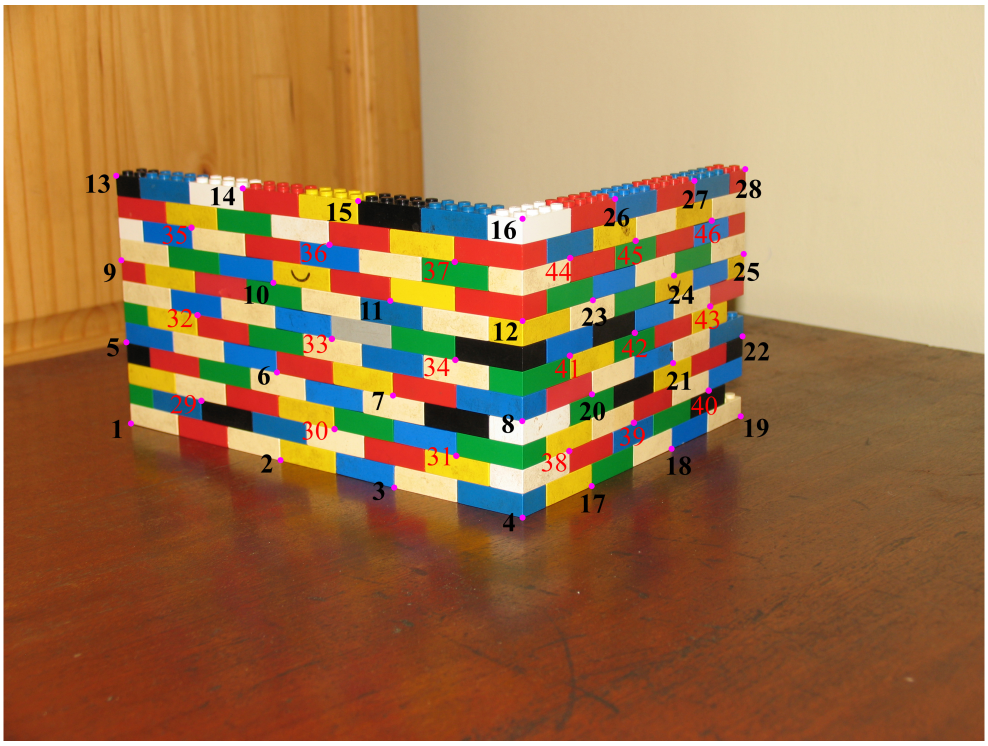


Figure 3.7: Calibration object with selected points

This decomposes as

$$\widetilde{C}_1 = \begin{bmatrix} -479.213 \\ -289.044 \\ 195.839 \\ 1 \end{bmatrix}$$

$$K_1 = \begin{bmatrix} 4286.706 & 5.017 & 1592.765 \\ 0 & 4303.529 & 1041.885 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_1 = \begin{bmatrix} 0.537 & -0.800 & -0.006 \\ -0.172 & -0.121 & -0.977 \\ 0.781 & 0.587 & -0.211 \end{bmatrix}.$$

For the second image of the calibration object, shown in Figure 3.8, the cal-

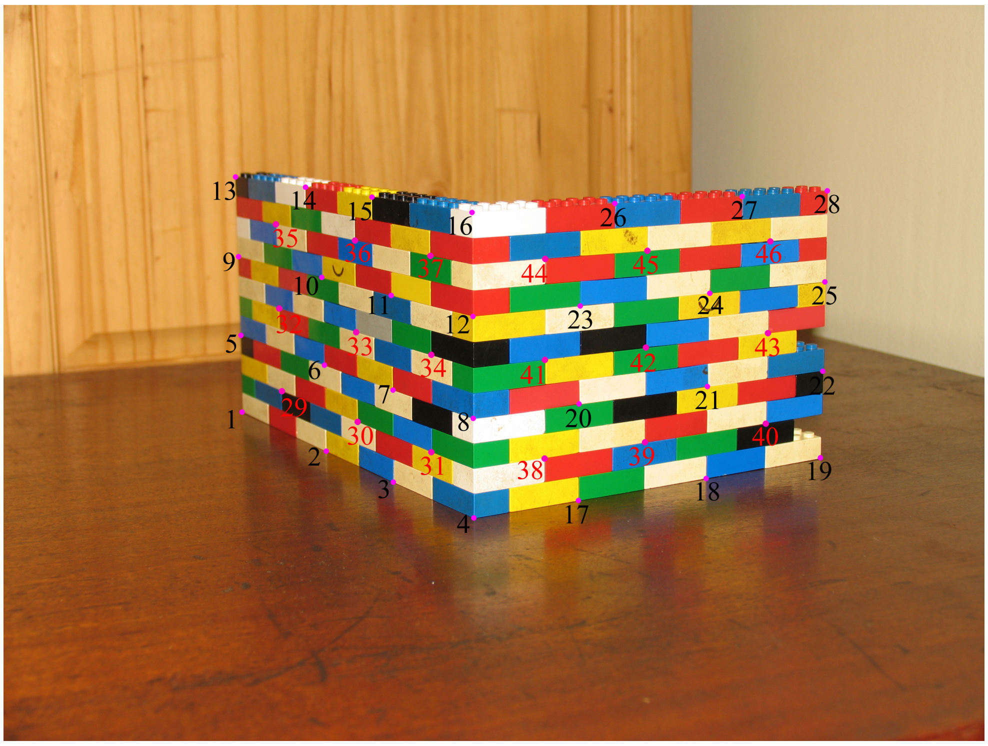


Figure 3.8: Calibration object with selected points

culated projection matrix is

$$P_2 = \begin{bmatrix} 7.973 & -1.232 & -0.487 & 1617.850 \\ 0.262 & 0.323 & -7.723 & 1800.115 \\ 0 & 0 & 0.003 & 1 \end{bmatrix}.$$

This decomposes as

$$\widetilde{C}_2 = \begin{bmatrix} -263.300 \\ -473.213 \\ 204.314 \\ 1 \end{bmatrix}$$

$$K_2 = \begin{bmatrix} 4265.707 & -8.638 & 1741.120 \\ 0 & 4280.402 & 1054.327 \\ 0 & 0 & 1 \end{bmatrix}$$

$$R_1 = \begin{bmatrix} -0.860 & 0.509 & -0.007 \\ 0.088 & 0.164 & 0.982 \\ -0.501 & -0.844 & 0.187 \end{bmatrix}.$$

The camera centers given by this decompositions seem to make sense. By looking at the images in Figure 3.7 and Figure 3.8 we can deduce that the camera was back from the origin in the negative direction on both the X and Y axes, while also being above the origin, which gives a positive Z values.

Looking at the camera matrix K_1, we compute the aspect ratio $\alpha_x/\alpha_y = 0.996$, with $K-2$ giving an aspect ratio of $\alpha_x/\alpha_y = 0.997$. An aspect ratio of approximately one indicates square pixels as we expect for a modern digital camera. This is confirmed by the value of the skew parameter in both K matrices, which is very small compared to α_x and α_y. From K_1 we see that the principal point is $[1592.765, 1041.885]^T$. As the image size is 3264×2448, this is close to the middle of the image as we would expect. The same can be said for the principal point in K_2, which is $[1741.120, 1054.327]^T$.

If we use the calculated camera projection matrix to project the known world coordinates onto a plane, we can calculate an error between the measured points and the new projected points. This is called the RMS pixel error and was calculated as 1.325pixels averag across all the points in Figure 3.7. A RMS error of 1.354 was computed for Figure 3.8. In Figure 3.9 we show the dialog created to input the calibration points. As we use the mouse to mark the points, the best we can expect to do is near to single pixel accuracy, thus 1.325 and 1.354 are considered minimal errors.

3.5.2 Planar Calibration

We test the planar calibration algorithm, described in Figure 3.10, using the sequence of images in Figure 3.10 and the code in *ThreeDVision.PlanarCalibration* function. The images are all 3264×2448 pixels. In this sequence the calibration grid was stationary with the camera at different orientations and positions, but the same effect can be achieved by keeping the camera stationary and moving the grid. The method gives the internal matrix as

$$TK_1 = \begin{bmatrix} 4309.193 & -19.634 & 1756.021 \\ 0 & 4367.197 & 1165.628 \\ 0 & 0 & 1 \end{bmatrix}$$

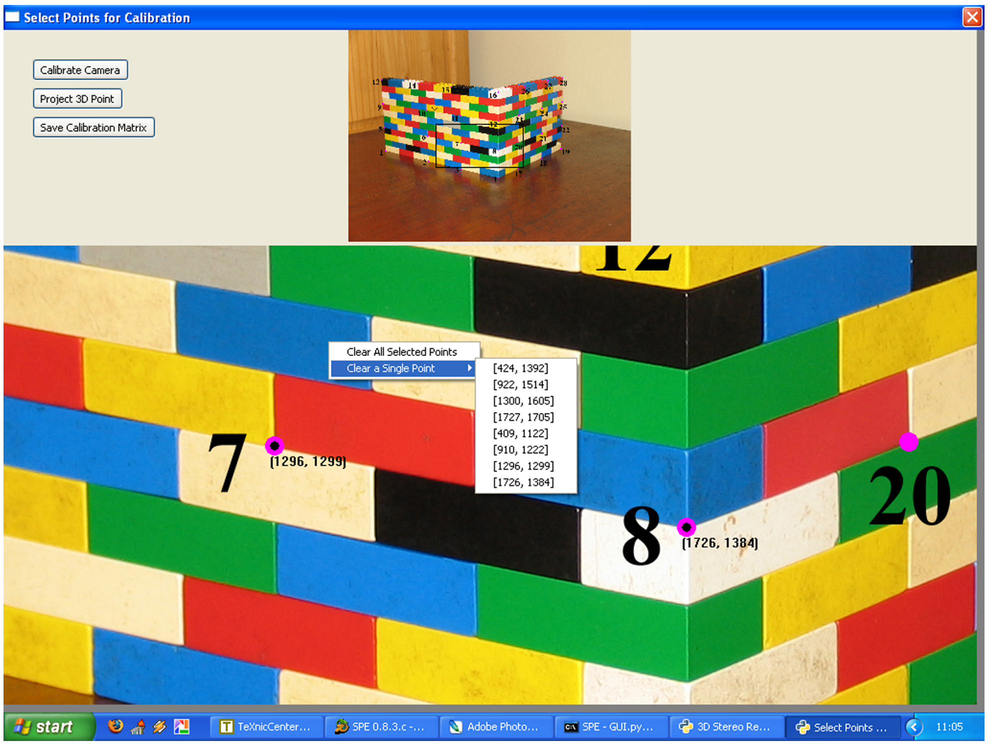


Figure 3.9: Dialog for selection of points

If we enforce the zero skew constraint we get the following result

$$K = \begin{bmatrix} 4327.446 & 0 & 1746.560 \\ 0 & 4388.351 & 1174.907 \\ 0 & 0 & 1 \end{bmatrix}$$

The difference between the K matrices calculated with the planar method and that of the linear method as percentages is given in Table 3.1. We notice that there is only a small difference between the focal lenghts of all the camera matrices, leading us to believe that they are indeed quite accurate. The slightly larger difference between the principal points of the linear and planar calibration can be attributed to the lack of radial distortion correction in our algorithm.

Although these differences might seem severe, it will be seen in the next chapter, when the matrices are used fo r3D reconstruction, that they do not render the algortihm unusable.

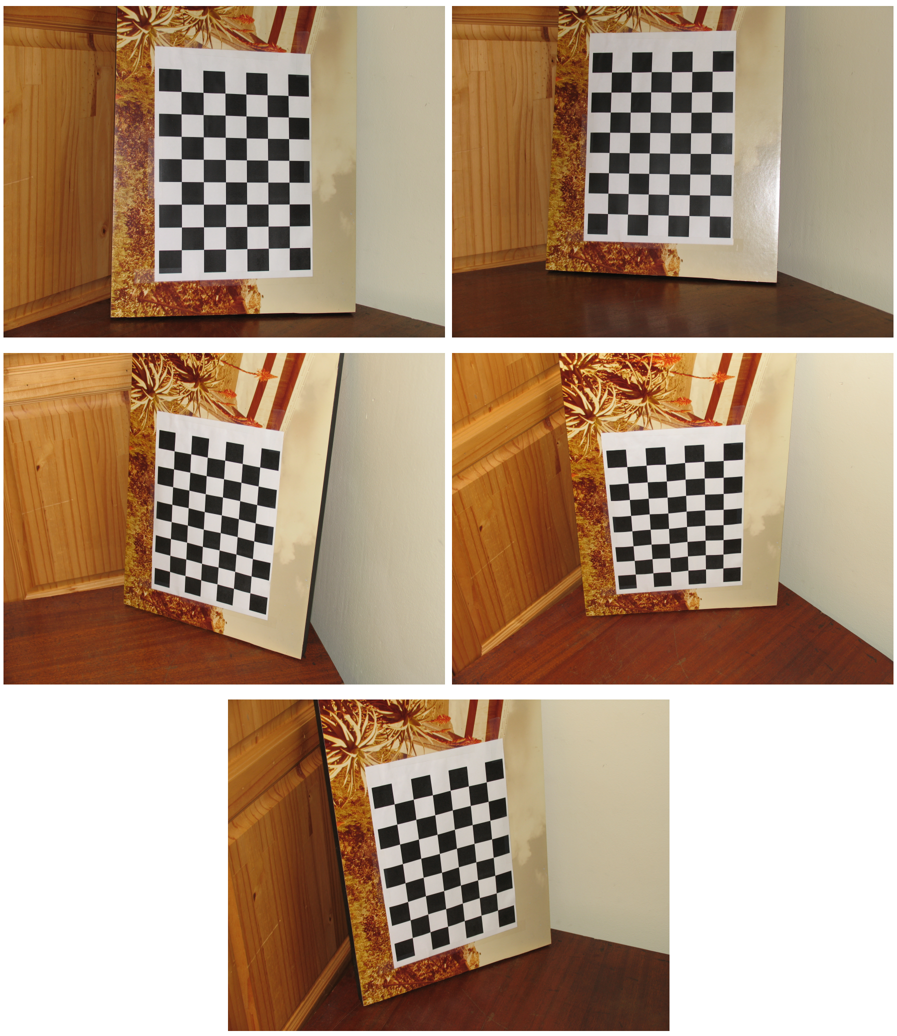


Figure 3.10: Sequence of images for planar calibration

Compared Matrices	α_x	α_y	p_x	p_y
K_1 and TK_1	0.522	1.458	9.296	10.615
K_1 and TK_2	0.941	1.932	8.805	11.321
K_2 and TK_1	1.009	1.987	0.848	9.548
K_2 and TK_2	1.426	2.459	0.311	10.262

Table 3.1: Differences between calibration matrices in percentages

Chapter 4

Stereo Vision

Almost all 3D reconstruction algorithms for computers are based on human vision. Like the eyes make images of the world and the brain calculates a 3D position, so the digital camera take photos and the computer algorithms do the mathematics. The details of human vision was first discovered in 1838 by Charles Wheatstone [33]. The field of research trying to mimic this process with computer is known as Stereo Vision and is based on the process of triangulation. In this chapter we explain the mathematical foundation for this process.

Throughout the chapter we use the notation given in *Multiple View Geometry in Computer Vision* [15]. Each of the images in a stereo pair have an associated camera matrix, denoted by P and P', where $'$ indicates all entities associated with the second view. A 3D point $\boldsymbol{X}$ will be imaged as $\boldsymbol{x} = P\boldsymbol{X}$ in the first image and as $\boldsymbol{x}' = P'\boldsymbol{X}$ in the second view. The points $\boldsymbol{x}$ and $\boldsymbol{x}'$ are called a correspondence pair as they are images of the same 3D point.

4.1 Epipolar Geometry

When two cameras view a 3D scene from two distinct positions, or a single camera takes two images sequentially whilst moving through the scene, there are a number of geometric relations between the two images. This intrinsic projective geometry is dependent only on the camera's internal parameters and the **relative** pose of the views. One such relation between the two images is called Epipolar Geometry.

In Figure 4.1 the 3D point $\boldsymbol{X}$ is imaged in two images with focal points at C and C'. The *epipole*, e and e' respectively, is the point where the line joining the two focal points (baseline) intersects the image plane. Any plane containing the baseline is part of the family (pencil) of *epipolar planes*. The *epipolar line* is the intersection of an epipolar plane with the image plane. Logically, if all epipolar planes contain the baseline, it means that all epipolar lines intersect at the epipole.

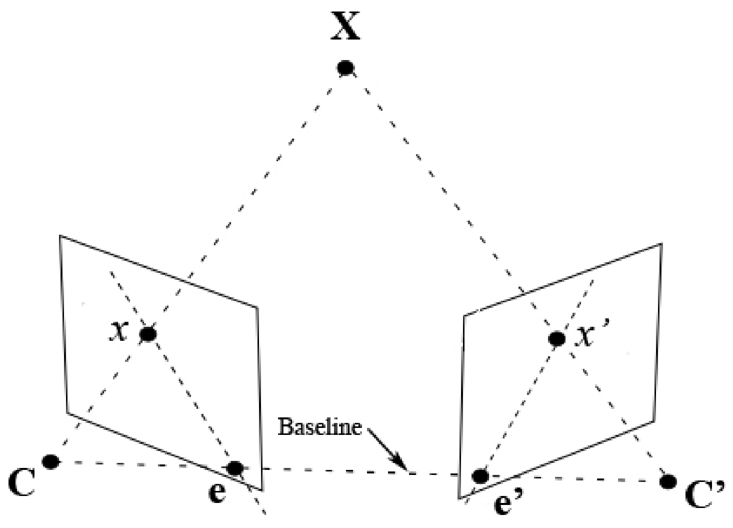


Figure 4.1: An Example of Epipolar Geometry

Two epipolar constraint become apparent if we study the image in Figure 4.1. Firstly, if we know the position of x in the image we can define the epipolar plane using the baseline and the point x. Where the epipolar plane intersects the other image plane gives us the other epipolar line, and thus defines a line on which x' must lie. This provides a constraint with which we can test if two points really correspond to the same 3D point. In terms of a stereo correspondence algorithm we can now search across a line instead of the whole image [24, 27].

The other constraint being if we know both points x and x' then their projection lines are also known. Seeing as they image the same 3D point their projection lines will intersect precisely at the point X. This constraint is used extensively in Section 4.4 for triangulation.

In Figure 4.2 two images are given with the epipolar lines of six points displayed in each. The six points are points 1,4,13,16,19 and 28. We can see that the epipoles are far from the image centers. The epipolar geometry was established using the camera matrices from linear calibration in section 3.5.1 and the method described in section 4.2.1.

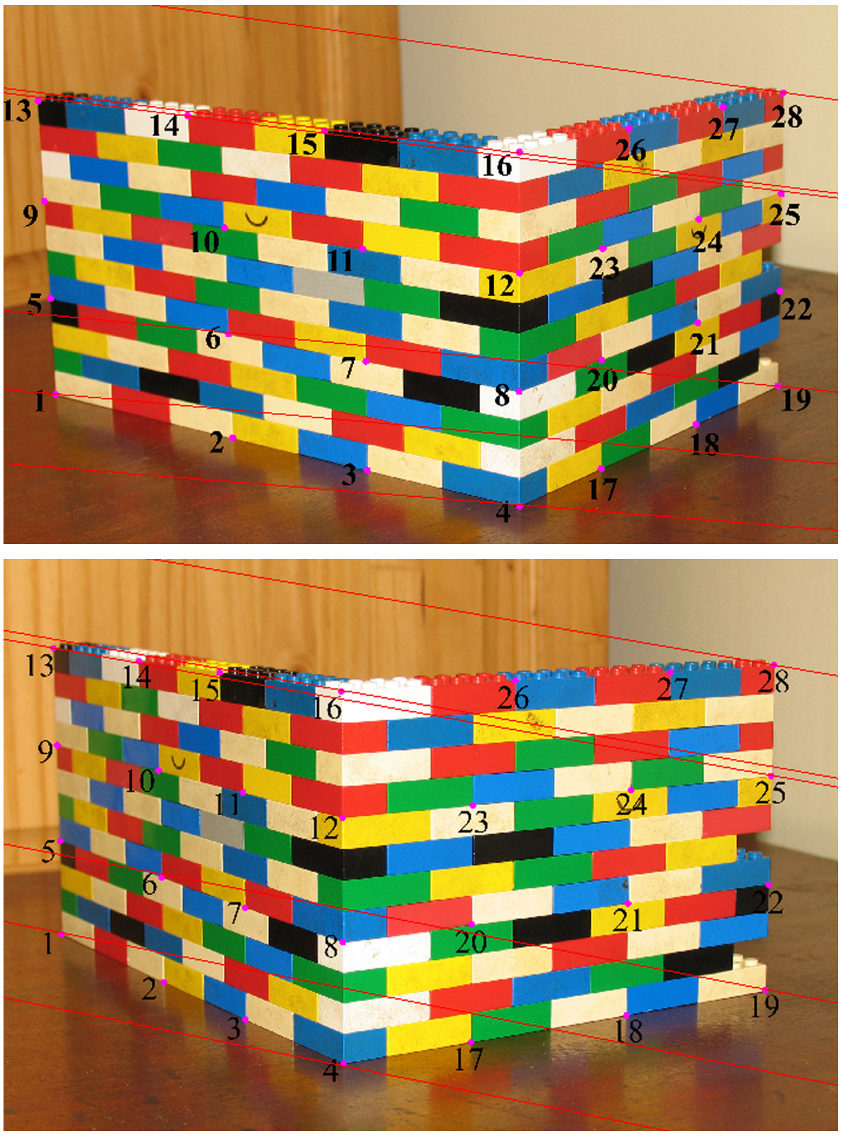


Figure 4.2: Example of Epipolar lines drawn in images

4.2 The Fundamental Matrix

The essential matrix (E) was introduced by Longuet-Higgins [17] to solve the relative placement problem for *calibrated* cameras. Although the essential matrix was proposed first, we will first decribe the more general fundamental matrix (F) which is defined for the uncalibrated case. The fundamental matrix is the unique 3×3 homogeneous matrix of rank 2 which satisfies

$$\boldsymbol{x}'^T F \boldsymbol{x} = 0$$

for all corresponding points $\boldsymbol{x} \leftrightarrow \boldsymbol{x}'$. It is a compact algebraic representation of the epipolar geometry of two images. The term *fundamental matrix* was first used by Q. T. Luong in his PhD thesis [18] in 1992.

We present some properties of the fundamental matrix here. For proofs of these properties, see [17, 11, 13].

If F is the fundamental matrix of a pair of cameras (P, P'), then:

1. F^T is the fundamental matrix for the pair in opposite order, (P', P).
2. If $\boldsymbol{x} \leftrightarrow \boldsymbol{x}'$ is a pair of matching points in the two images, then $\boldsymbol{x}'^T F \boldsymbol{x} = 0$.
3. If $\boldsymbol{x}$ is a point in the first image, then $F\boldsymbol{x}$ is the epipolar line in the second image.
4. The matrix F has rank two.
5. The epipole $\boldsymbol{e}'$ is the left null-vector of F and the right null-vector gives us $\boldsymbol{e}$.

Figure 4.3: Properties of the fundamental matrix

Although there are a multitude of methods to compute F, depending on the available information, they all stem from two main algorihtms. One where we have two images and their camera matrices, P and P', readily available and the other which requires only point correspondences, $\boldsymbol{x} \leftrightarrow \boldsymbol{x}'$, called the 8-point algorithm.

4.2.1 Computation of F from two camera matrices

The following algebraic derivation of the computation of F from two camera matrices is given by Xu and Zhang [37].

We are given two camera projection matrices, P and P'. The ray back-projected from $\boldsymbol{x}$ by P is obtained by solving $P\boldsymbol{X} = \boldsymbol{x}$. The one-parameter family of solutions is then given by

$$\boldsymbol{X}(\lambda) = P^{+}\boldsymbol{x} + \lambda\widetilde{\boldsymbol{C}}$$

where P^{+} is the pseudo-inverse of P, i.e. $PP^{+} = I$, and $\widetilde{\boldsymbol{C}}$ is the camera center. Two points of interest on the ray are $P^{+}\boldsymbol{x}$ (at $\lambda = 0$) and the first camera center $\widetilde{\boldsymbol{C}}$ (at $\lambda = \infty$). These two points are imaged by the second camera P' as $P'P^{+}\boldsymbol{x}$ and $P'\widetilde{\boldsymbol{C}}$ respectively. The epipolar line is the line joining these two projected points, namely $\boldsymbol{l'} = (P'\widetilde{\boldsymbol{C}}) \times (P'P^{+}\boldsymbol{x})$. The point $P'\widetilde{\boldsymbol{C}}$ is the epipole in the second image and is denoted by $\boldsymbol{e'}$. Thus, $\boldsymbol{l'} = [\boldsymbol{e'}]_{\times}(P'P^{+})\boldsymbol{x} = F\boldsymbol{x}$, where F is then the matrix

$$F = [\boldsymbol{e'}_{\times}]P'P^{+}. \tag{4.2.1}$$

The matrix F computed by this method must satisfy the condition that for any pair of corresponding points $\boldsymbol{x} \leftrightarrow \boldsymbol{x'}$ in the two images

$$\boldsymbol{x'}^{T}F\boldsymbol{x} = 0. \tag{4.2.2}$$

This equation encapsulates the epipolar geometry described in the previous section and leads us to the next method of computing F.

4.2.2 Normalised 8-Point Algorithm

The 8-point algorithm is one of the simplest, linear methods of computing F from only point correspondences. If proper normalisation of the input data is implemented and care is taken in the measurements, the algorithm performs surprisingly well. The algorithm was first presented by Longuet-Higgins in 1981 [17].

The algorithm is in many ways analogous to the DLT algorithm (refer to Section 3.1), including the reasons for and method of normalisation. The biggest difference being the enforced singularity condition before the denormalisation step. A justification of this is presented in detail by Hartley in

[12].

If we view a single point correspondence $\{\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}'_i\}$ as $\boldsymbol{x} = (x, y, 1)^T$ and $\boldsymbol{x}' = (x', y', 1)^T$ then according to equation (4.2.2) we have one constraint on F for each pair. Specifically, the constraint can be written as

$$(x'x, x'y, x', y'x, y'y, y', x, y, 1)\boldsymbol{f} = 0$$

where $\boldsymbol{f}$ is the 9-vector made up of the entries of F in row-major order. From a set of n point correspondences we create a set of linear equations of the form

$$A\boldsymbol{f} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \boldsymbol{f} = \boldsymbol{0}. \qquad (4.2.3)$$

The algorithm as given by Hartley and Zisserman [15] is summarised in Figure 4.4.

Objective

Given $n \geq 8$ image point correspondences $\{\boldsymbol{x}_i \leftrightarrow \boldsymbol{x}'_i\}$, determine the fundamental matrix such that $\boldsymbol{x}'_i F \boldsymbol{x}_i = 0$.

Algorithm

1. **Normalisation**: Transform the image coordinates according to $\hat{\boldsymbol{x}}_i = T\boldsymbol{x}_i$ and $\hat{\boldsymbol{x}}'_i = T'\boldsymbol{x}'_i$ such that the centroid is at the origin and the average distance from the origin is $\sqrt{2}$.
2. **Linear Solution**: Determine $\widehat{F}$ from the singular vector corresponding to the smallest singular value of $\widehat{A}$, where $\widehat{A}$ is composed as in (4.2.3).
3. **Constraint enforcement**: Replace $\widehat{F}$ by $\widehat{F}'$ such that $\det \widehat{F}' = 0$ using the SVD.
4. **Denormalisation**: Set $F = T'^{-T}\widehat{F}'T$. Matrix F is the fundamental matrix of the original data.

Figure 4.4: The Normalised 8-point algorithm for F.

4.2.3 Extraction of cameras from Fundamental Matrix

From the properties of the fundamental matrix in Figure 4.3 we can deduce that F depends only on the choice of image coordinate frame, and not on the choice of the world coordinate frame. In fact the fundamental matrix is unchanged by a projective transformation of 3-space, even though this would change the cameras, P,P'. For example if H is a 4×4 matrix representing a projective transformation, then the fundamental matrices corresponding to the pairs of camera matrices (P, P') and $(PH, P'H)$ are the same.

From equation (4.2.1) we see that a pair of cameras defines a unique matrix F, but from the paragraph above we conclude that the converse is not true. Camera matrices can only be determined up to a projective transformation by the fundamental matrix. Given this ambiguity we define a specific *canonical form* for the pair of camera matrices from a given F. For the first camera we have the simple form $P = [I|\mathbf{0}]$, where I is a 3×3 identity matrix and $\mathbf{0}$ is a null 3-vector. The other camera is given as $P' = [M|\boldsymbol{m}]$ which makes $F = [\boldsymbol{m}]_\times M$.

To define exactly what P' should be, we will use a characterisation of the fundamental matrix given by Hartley and Zisserman [15] as:

> A non-zero matrix F is the fundamental matrix corresponding to a pair of camera matrices P and P' if and only if $P'^T FP$ is skew-symmetric.
>
> **Proof**. The condition that $P'^T FP$ is skew-symmetric is equivalent to $\boldsymbol{X}^T P'^T FP\boldsymbol{X} = 0$ for all $\boldsymbol{X}$. Setting $\boldsymbol{x} = P\boldsymbol{X}$ and $\boldsymbol{x'} = P\boldsymbol{X'}$, this is equivalent to $\boldsymbol{x'}^T F\boldsymbol{x} = 0$, which is the definition of a fundamental matrix.

Now if we let S represent any skew-symmetric matrix, we have

$$P = [I|\mathbf{0}] \text{ and } P' = [SF|\boldsymbol{e'}],$$

where $\boldsymbol{e'}$ is the epipole such that $\boldsymbol{e'}^T F = 0$. As the final step, we define S as $S = [\boldsymbol{e'}]_\times$, as is suggested by Luong and Viéville [20], which then gives us:

$$P = [I|\mathbf{0}] \text{ and } P' = [[\boldsymbol{e'}]_\times F|\boldsymbol{e'}]. \tag{4.2.4}$$

The algorithm is summarised in Figure 4.5.

Objective

Given a fundamental matrix F, determine the camera matrices of the two views.

Algorithm

1. Find the two epipoles, $\boldsymbol{e}$ and $\boldsymbol{e'}$, by using the SVD of F. See the last point in Figure 4.3.
2. Set the first camera as $P = [I|\mathbf{0}]$.
3. Create skew-symmetric matrix with

$$S = \begin{bmatrix} 0 & -e'_3 & e'_2 \\ e'_3 & 0 & -e'_1 \\ -e'_2 & e'_1 & 0 \end{bmatrix}$$

   where $\boldsymbol{e'} = [e'_1, e'_2, e'_3]^T$.
4. Now set the second camera to $P' = [SF|\boldsymbol{e'}]$

Figure 4.5: Retrieving the camera matrices.

4.3 The Essential Matrix

As has been mentioned, the essential matrix E is a specialisation of the fundamental matrix. The fundamental matrix is more general in that it removes the assumption of calibrated cameras. However, if the internal parameters K of the cameras P and P' are available, we can use normalised coordinates.

4.3.1 Normalised coordinates

If we consider the equations $\boldsymbol{x} = P\boldsymbol{X}$ and $P = K[R|\boldsymbol{t}]$. We can now apply the inverse of K to the image point $\boldsymbol{x}$ to obtain $\hat{\boldsymbol{x}} = K^{-1}\boldsymbol{x}$. Then we have *normalised coordinates* $\hat{\boldsymbol{x}}$ as in $\hat{\boldsymbol{x}} = [R|\boldsymbol{t}]\boldsymbol{X}$. We also define the camera matrix as $K^-1P = [R|bmt]$ where we have removed the effect of the known calibration matrix K.

The defining equation for the essential matrix is similar to that for the fundamental matrix.

$$\hat{\boldsymbol{x}}'^T E \hat{\boldsymbol{x}} = 0.$$

If we substitute $\hat{\boldsymbol{x}}$ and $\hat{\boldsymbol{x}}'$ we get $\boldsymbol{x'}^T K'^{-T} E K^{-1} \boldsymbol{x} = 0$. Comparing this to equation (4.2.1) we find the relationship between the essential matrix and fundamental matrix is

$$E = K'^T F K. \tag{4.3.1}$$

4.4 Triangulation

Now that we have the mathematical tools to describe and compute the epipolar geometry of two views, we move on to the subject of using them. The process of computing a 3D point given its image in two views is called triangulation.

In theory, the triangulation problem is trivial. As seen in Figure 4.6, each point in an image has an associated ray (called its back-projected ray). These rays intersect at a common point, which is the original 3D point. In practice, however, the coordinates can never be measured to such infinite accuracies. Instead, there are almost always measurement and distortion errors, which mean that the rays will not intersect perfectly, as seen in Figure 4.7. In such as case, we have to estimate the best solution. Throughout this section we assume that any errors (noise) are only in the measured image coordinates, not in the camera matrices, P and P', or in their intrinsic parameters, K and K'.

4.4.1 Linear triangulation methods

The linear triangulation described here is a direct analogue of the DLT method given in Section 3.1. We use the two imaged points, $\boldsymbol{x} = P\boldsymbol{X}$ and $\boldsymbol{x'} = P'\boldsymbol{X}$, to form a linear equation in $\boldsymbol{X}$. If we write the cross-product, $\boldsymbol{x} \times (P\boldsymbol{X}) = \boldsymbol{0}$ for each point as

$$\begin{aligned} x(\boldsymbol{p}^{3T}\boldsymbol{X}) - (\boldsymbol{p}^{1T}\boldsymbol{X}) &= 0 \\ y(\boldsymbol{p}^{3T}\boldsymbol{X}) - (\boldsymbol{p}^{2T}\boldsymbol{X}) &= 0 \\ x(\boldsymbol{p}^{2T}\boldsymbol{X}) - y(\boldsymbol{p}^{1T}\boldsymbol{X}) &= 0 \end{aligned}$$

where $\boldsymbol{p}^{iT}$ are the rows of P, then we have three equations in $\boldsymbol{X}$ of which two are linearly independent. These equations can then be combined to form an

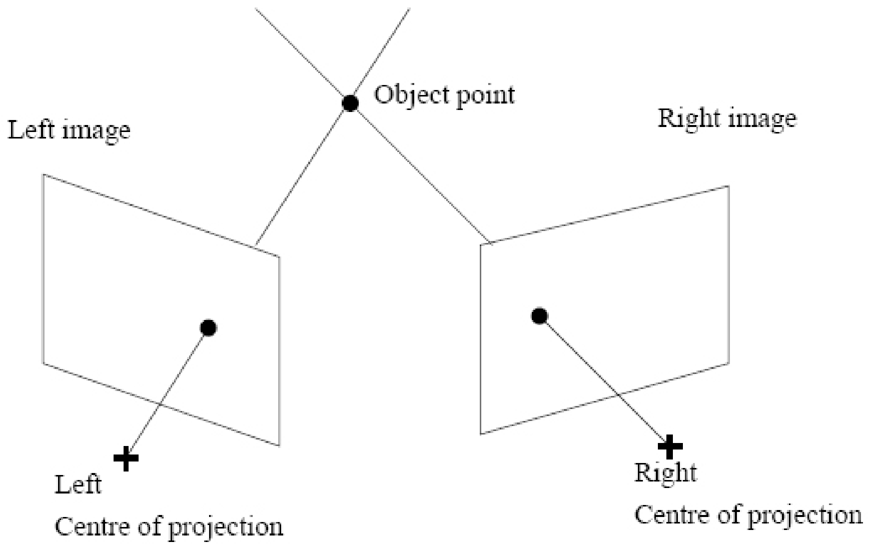


Figure 4.6: The principal of Triangulation

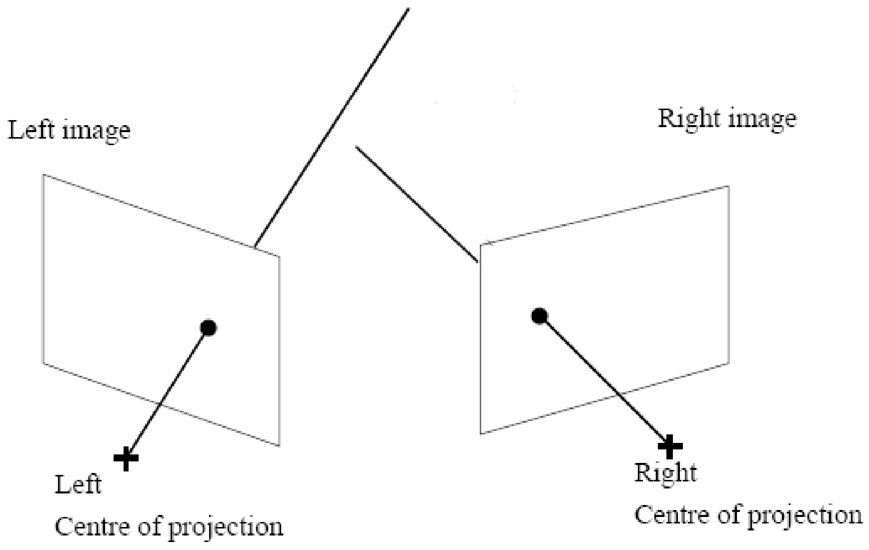


Figure 4.7: Back-projected rays from imperfectly measured points

equation of the form $A\boldsymbol{X} = \mathbf{0}$ with

$$A = \begin{bmatrix} x\boldsymbol{p}^{3T} - \boldsymbol{p}^{1T} \\ y\boldsymbol{p}^{3T} - \boldsymbol{p}^{2T} \\ x'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{1T} \\ y'\boldsymbol{p}'^{3T} - \boldsymbol{p}'^{2T} \end{bmatrix}$$

where two equations have been included from each view. Once again, as in the DLT method, we find the solution as the unit singular vector corresponding to the smallest singular value of A (see Appendix B).

4.4.2 Sampson Correction

As mentioned in the previous section and shown in Figure 4.7, there are errors in the calculation of the 3D point. The points calculated with the linear method will not satisfy the epipolar constraint,$\boldsymbol{x}'^T F\boldsymbol{x} = 0$. Sampson correction uses the epipolar constraint as a *cost function* to minimise the overall error.

If we measure the geometric distance between the estimated and measured image coordinates we have the *geometric error* given by

$$\sum_i d(\boldsymbol{x}_i, \hat{\boldsymbol{x}}_i)^2 + d(\boldsymbol{x}'_i, \hat{\boldsymbol{x}}'_i)^2 \text{ subject to } \hat{\boldsymbol{x}}'^T_i F\hat{\boldsymbol{x}}_i = 0 \text{ for all } i. \tag{4.4.1}$$

The geometric error, however, is quite complex in nature as it requires the minimisation of both the fundamental matrix and the points $\hat{\boldsymbol{x}}_i$ and $\hat{\boldsymbol{x}}'_i$ simultaneously. This leads to a computationally intensive non-linear estimation problem. In contrast with this is the *algebraic distance*, given by

$$d_{alg}(\boldsymbol{x}_1, \boldsymbol{x}_2)^2 = a_1^2 + a_2^2 \text{ where } \boldsymbol{a} = (a_1, a_2, a_3)^T = \boldsymbol{x}_1 \times \boldsymbol{x}_2 \tag{4.4.2}$$

for any two vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$. This cost function has a linear and thus unique solution and is computationally cheap. The disadvantage, however, is that the distance that is minimised is not really geometrically or statistically meaningful.

There is another cost function that lies between the geometric and algebraic error in terms of complexity and gives a good approximation of the geometric error. Sampson first used this approximation for conic fitting [26] and it has since become know as the *Sampson error*. The derivation of the

Sampson cost function is presented in full in *Multiple View Geometry* by Hartley and Zisserman [15], we only present the result as

$$\sum_i \frac{(\mathbf{x}_i'^T F \mathbf{x}_i)^2}{(F\mathbf{x}_i)_1^2 + (F\mathbf{x}_i)_2^2 + (F^T\mathbf{x}_i')_1^2 + (F^T\mathbf{x}_i')_2^2} \tag{4.4.3}$$

where $(F\mathbf{x}_i)_j^2$ represents the square of the j-th entry of the vector $F\mathbf{x}_i$. The approximation gives good results if the errors are small compared to the measurements. The reason the Sampson distance is much faster to compute than the geometric error is that equation (4.4.3) only involves parameters of F.

Now that we have defined a cost function we can compute the correction that needs to be made to the measured points to minimise the error. The Sampson correction δ_Z to the measured point $\mathbf{Z} = (x, y, x', y')^T$ is given as

$$\widehat{\mathbf{Z}} = \mathbf{Z} + \delta_Z = \mathbf{Z} - J^T(JJ^T)^{-1}\boldsymbol{\epsilon}.$$

The first-order approximation to the image point is then given as

$$\begin{pmatrix} \hat{x} \\ \hat{y} \\ \hat{x}' \\ \hat{y}' \end{pmatrix} = \begin{pmatrix} x \\ y \\ x' \\ y' \end{pmatrix} - \frac{\mathbf{x}_i'^T F \mathbf{x}_i}{(F\mathbf{x}_i)_1^2 + (F\mathbf{x}_i)_2^2 + (F^T\mathbf{x}_i')_1^2 + (F^T\mathbf{x}_i')_2^2} \begin{pmatrix} (F^T\mathbf{x}')_1 \\ (F^T\mathbf{x}')_2 \\ (F\mathbf{x})_1 \\ (F\mathbf{x})_2 \end{pmatrix}.$$

It should be noted that this is not an optimal method as the estimated points still do not exactly satisfy the epipolar constraint. An optimal method is given by Hartley and Sturm [14].

4.5 Results of Implementation

It is hard to visualise the results of 3D reconstruction. In Figure 4.8 we show the result of triangulation using the camera matrices obtained from linear calibration (see Section 3.5.1). The average Euclidean error of the reconstruction across the 46 points used, was 0.372mm. The reconstructed points are plotted as circles in Figure 4.8, while the ground truth is given as blue stars. On this scale, such small errors are almost unnoticeable. Therefore we will only give the average Euclidean error as an indication of accuracy further on. Table 4.1 contains a comparison between the three different methods used and how they performed on certain sets of points for calibration and reconstruction.

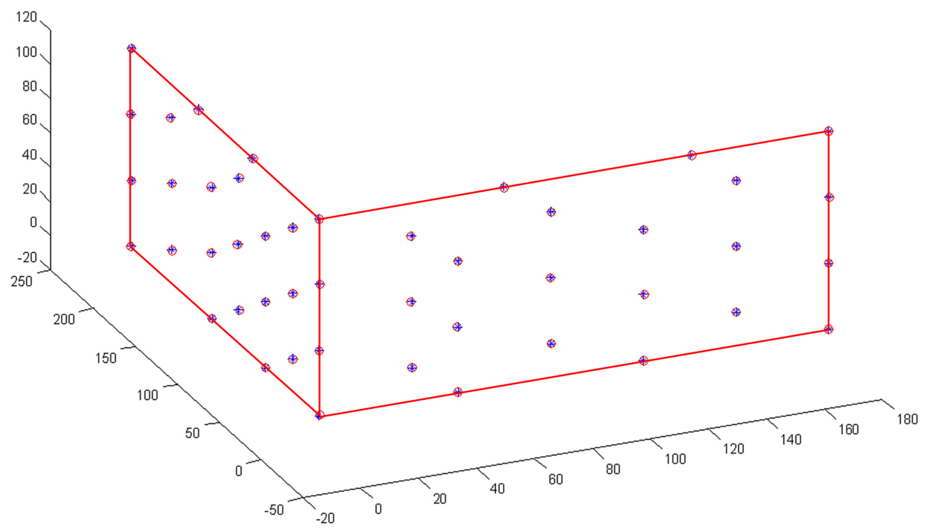


Figure 4.8: Triangulation using linear calibration camera matrices

It might appear that the linear calibration algorithm is the best choice, as it gives the best results, but we need to take the amount of information available to each algortihm into account. The linear algorithm uses two fully calibrated cameras, in other words both the camera projection matrices are available. The reconstructed points from this algorithm are automatically in the correct coordinate system as the geometric positions of the cameras are contained within the matrices P_1 and P_2.

Calibration	**Reconstructed**	**Linear**	**8-point**	**Essential Matrix**
Points 6-23	Points 6-23	0.3077	0.4208	0.5760
Points 6-23	Points 29-46	0.3808	0.4874	0.6980
Points 6-23	All 46 Points	0.3725	0.6951	0.8393
All 46 Points	All 46 Points	0.3725	0.4183	0.5140

Table 4.1: Results of Implementation

The Normalised 8-point method uses only point correspondences to calculate the relative positions of the two cameras from which triangulation can be performed. We require a minimum of 8 point correspondences to compute F. Because the Euclidean positions of the cameras are not available, the reconstructed points are calculated only up to a projectivity. We need to use a 3D-to-3D homography to ugrade the reconstruction to metric. At least 8 points with known world coordinates are required to compute the homography.

The last method used involved computing the essential matrix from point correspondences. A minimum of 6 points and the internal parameters of the camera are required to calculate the essential matrix. This time the points are reconstructed up to a similarity. To upgrade the reconstruction to metric we need at least 3 known world coordinates. These are then used to calculate the rotation, translation and scale needed to move the points to the correct coordinate system. This is the method used by Photomodeler®.

The results of the essetial matrix method were slightly worse that those of the normalised 8-point algortihm. This can be attributed to the incomplete implementation of the Zhang calibration algorithm. With a full implementations we would expect better results.

An intresting observation in all the reconstructions performed was that the errors in the two dimensions parallel to the image plane were similar, but thet the errors in the depth field were much larger. This was confirmed when the system used by Bhukka (Pty) Ltd was investigated. It is thus very important to place targets or control points in such a manner to get a good estimate of depth.

4.6 Conclusions

In this chapter we explained the mathematics that allows reconstruction from stereo vision. The different methods to calculate the fundamental matrix (and essential matrix) were explained in detail and with a simple implementation we showed that a photogrammetry system can perform accurate point reconstruction given a wide vareity of information.

In the next chapter we investigate an actual system used by Bhukka (Pty) Ltd in their operations and propose a target extraction algorithm.

Chapter 5

Target Extraction

Bhukka (Pty) Ltd use Photomodeler®[1] as part of the system that they use for photogrammetry. Only the center lines of pipes are modeled using Photomodeler®after which the data is exported to MicroStation from Bentley (Inc) for further processing. Photomodeler®uses a planar calibration algorithm for camera calibration. Bhukka (Pty) Ltd have chosen to make use of targets in the reconstruction process, to increase accuracy.

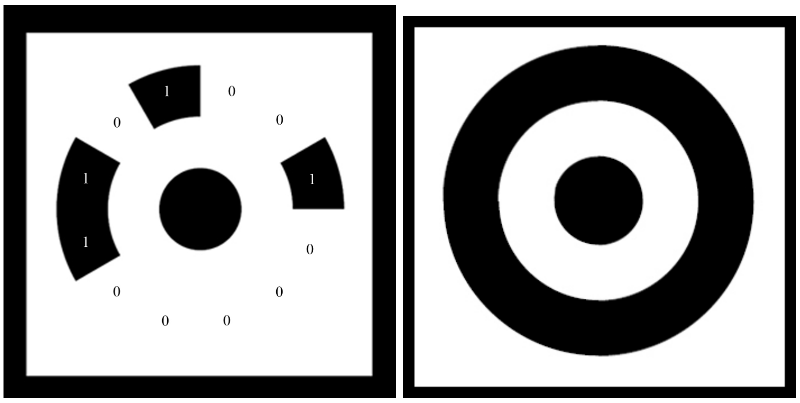
The use of coded targets in 3D reconstruction is not uncommon and a good review is provided by Ahn *et al.* [1]. The automation of the measurement process is an important step from the lab research toward a practical system that serves as part of a production flow. The target extraction within Photomodeler®is not robust and very slow to use. In this chapter we describe an algorithm to automatically extract targets form images, identify the coded targets and then load the images along with the extracted data into PhotoModeler®.

5.1 Coded Targets

The targets used by PhotoModeler® are high contrast circular targets placed in a scene to provide an accurate sub-pixel point marking. Around the central disc of the target are some additional bits that can be automatically recognised during the marking process. In Figure 5.1(a) the coded target with binary code 100101100000 is shown. Targets all have a unique 12-bit binary code associated with them, where a black bit denotes a one and a blank bit a

[1]See http://www.photomodeler.com/ for more information

zero. The start bit is at three o'clock in Figure 5.1(a) where the first bit clockwise from there is always a zero and the first bit anti-clockwise is always a one. The reason for using circular targets is to remove the angular constraints of a non-circular target. It does not matter at what angle the circular target is put up, in the image the code will appear the same. In Section 5.3 we describe how we read these codes from the images. If the target has a continous



(a) Coded Target (b) 'Dummy' Target

Figure 5.1: Example of a coded and 'dummy' target

ring as a code it is a 'dummy' target. These targets are not surveyed and are only there to provide more corresponding points between images. They are also marked, but have no associated world coordinates. An example of a 'dummy' target is given in Figure 5.1(b).

5.2 Target Extraction

Once we have images with targets in them we need to determine the position of the center of each target. A typical image with such targets is given in Figure 5.2.

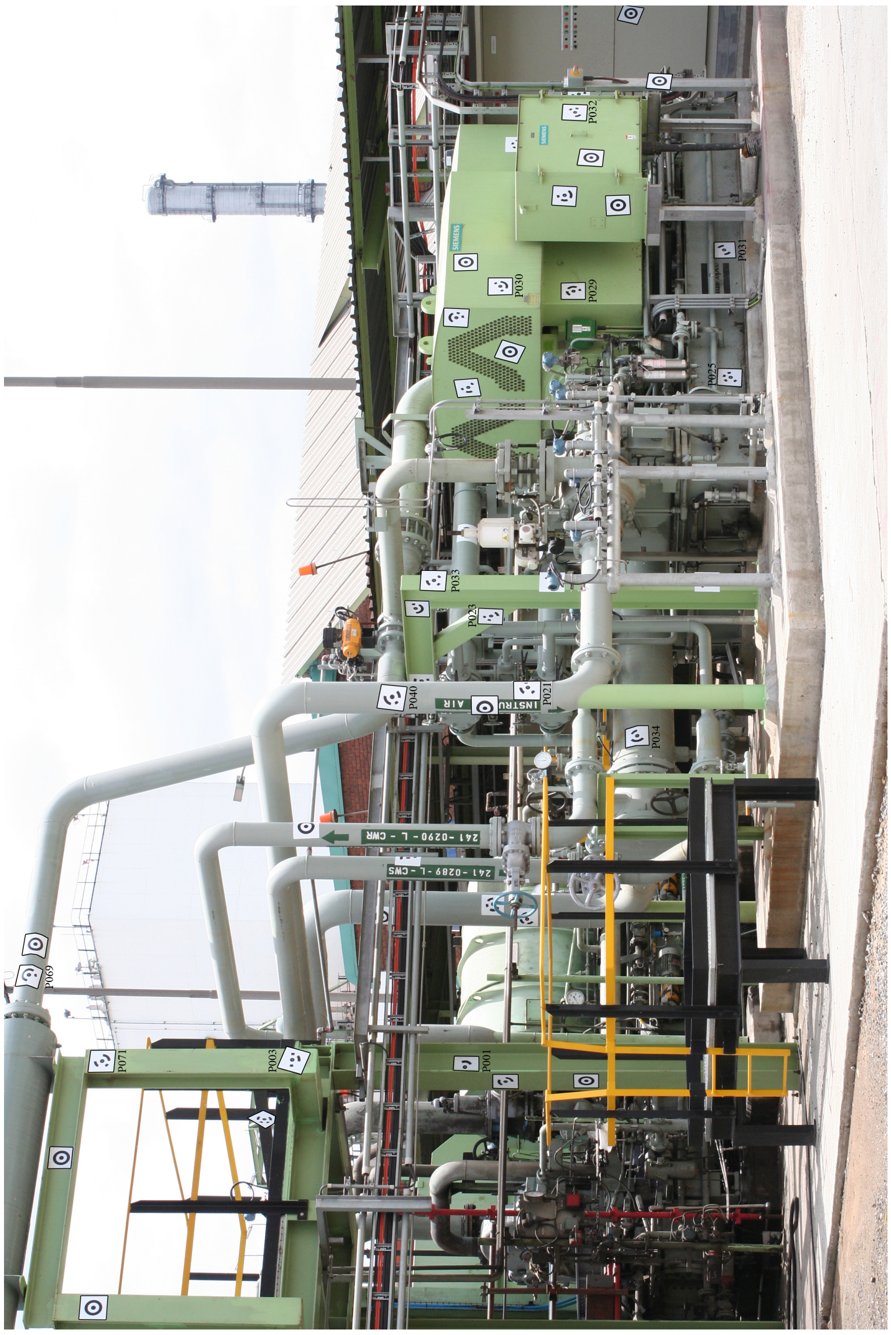


Figure 5.2: Example of targets in image

We start by splitting the RGB image into its three channels, so we have three separate images, one each for red, green and blue. We are looking for the white squares of the targets in the image. White contains all colours and so any white pixels should have high values in all three images. We therefore threshold all the values in the three images and keep only those above 50% of the highest brightness in the image. An example of the green channel before, Figure 5.3, and after thresholding, Figure 5.4 is given here.

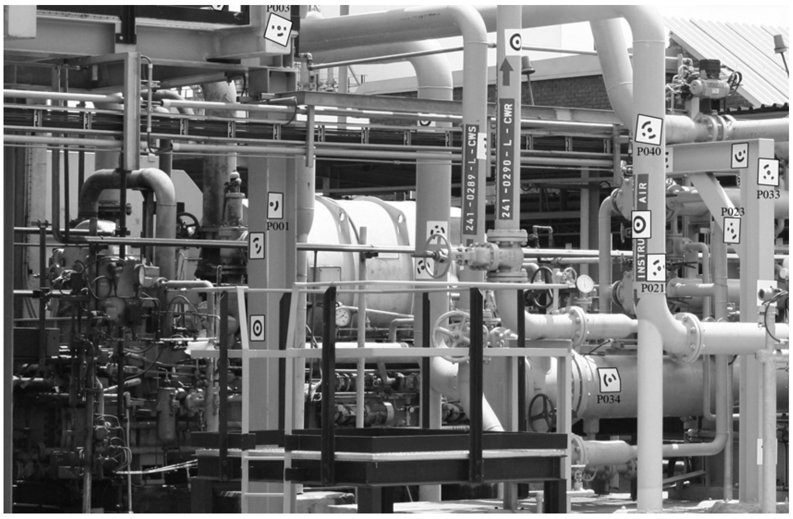


Figure 5.3: The green channel before thresholding

White areas will not only have high values, but will also have almost the same value in all three spectra. To use this property we measure the absolute difference between all three images and keep only those with a variance smaller than 15%. Next, we merge the three layers back into one greyscale image, see Figure 5.5. Our next step is to open the image by first eroding and then dilating using a 3×3 structuring element. After which we have the image displayed in Figure 5.6. Note that much of the smaller white artifacts have been removed, which improves the efficiency of the next step considerably. Following the opening of the image, we can run the contour detection

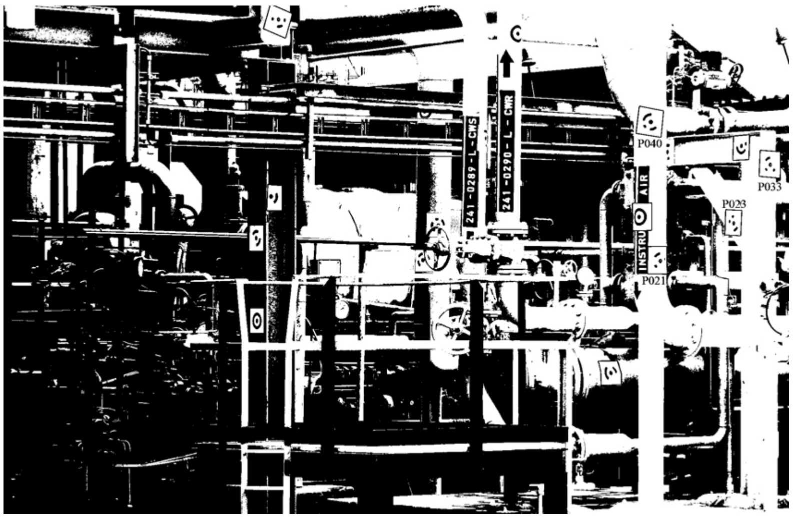


Figure 5.4: The green channel after thresholding

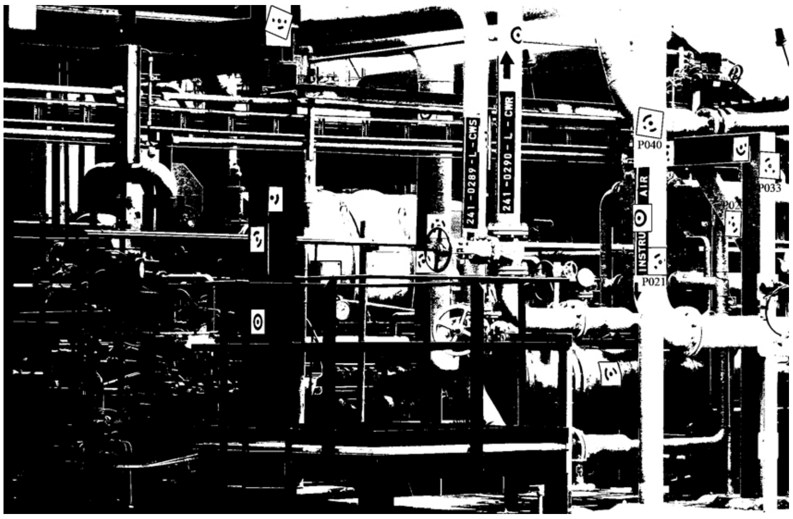


Figure 5.5: Greyscale image after three channels are merged

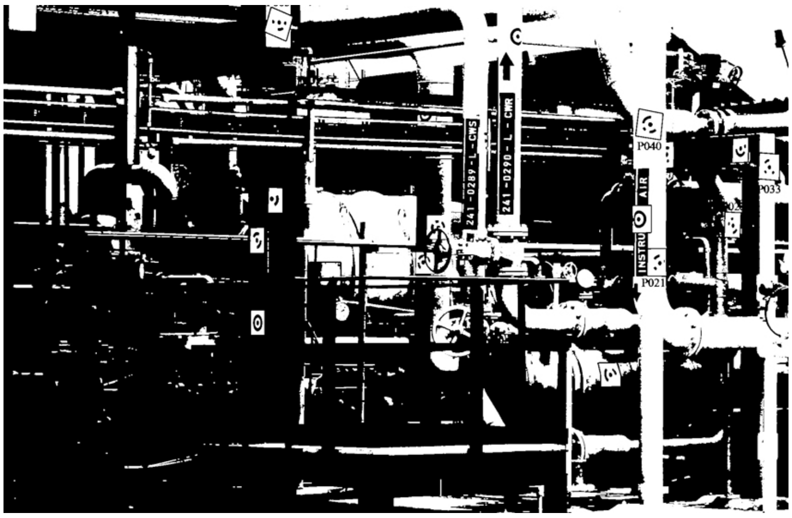


Figure 5.6: The sample image after opening, using erosion and then dilation

algorithm on the greyscale image. The results are shown in Figure 5.7 where red shows external contours and green the internal contours (holes). Next we

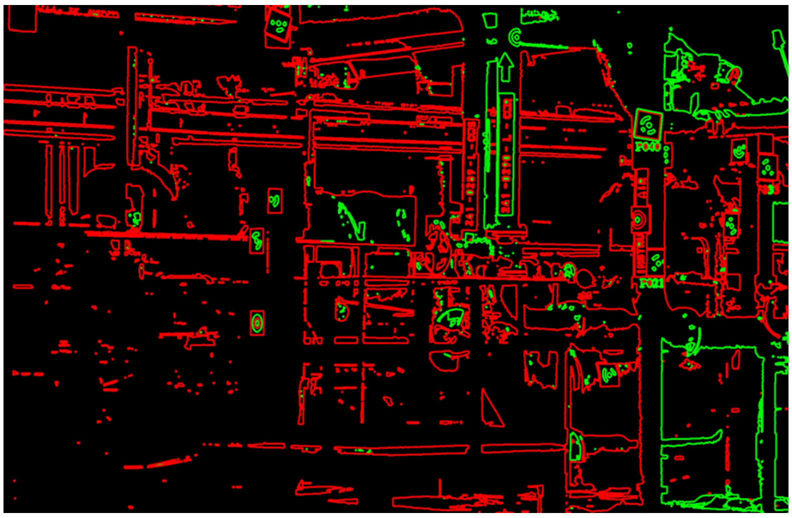


Figure 5.7: Contour detection

go through all the contours we found and try to identify possible targets. We do this by first approximating all contours as polygonal curves by use of the OpenCV implementation of the Douglas-Peucker algorithm [6]. Now we test a couple of thing to determine if we have a target or not. Specifically a square should have 4 vertices after approximation, be convex, not be a hole and have a certain size relative to the image. We use OpenCV CheckContourConvexity function to check whether the contour is convex.

We know that a target must have at least 3 holes in it and at most 10, so any square outside that limit is not a target. For now we assume that all the squares we are left with are in fact targets. With the outer most contour available, we guess that the center of the target will be somewhere near the center of this contour. To refine this estimate we need to find the circle in the middle of the target and fit an ellipse over it. The center of this ellipse is then the accurate center of the target. To find the middle circle we look at all the contours inside the square and find the smallest one that contains our center guess, in other words, the guess is inside the contour.

5.3 Target Identification

Now that we have the center of the target and the outline we need to identify the target code. Before we do this however, we map the target (contained in the contour on the original image) to a frontal view using a 2D-to-2D homography and then threshold so we have a black and white image. We map each of the corners to a 50 $\times$ 50 blank image and the result is shown in Figure 5.8. Once we have a frontal view of the target, we use Bresenham's line algorithm

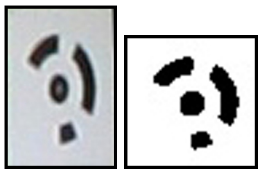


Figure 5.8: Result of the homography applied to the target

[4] to draw lines from the center of the target to the outer contour. The lines are evenly spaced on the outer contour, as shown in Figure 5.9, and we measure the times that the colour changes along each line to see if it is a zero or one bit. When we have the binary code, we shift it until we find a matching code in the database, making sure that the first bit is always a one and the last bit is a zero. If the target has twelve one bits, or 111111111111, then it is a 'dummy' target.

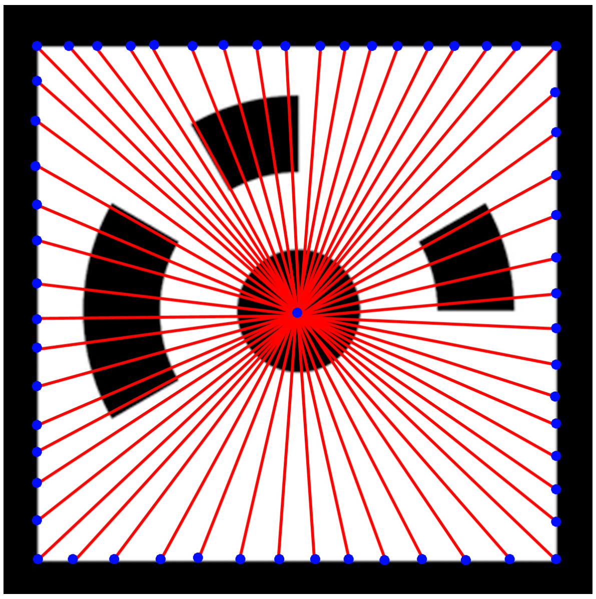
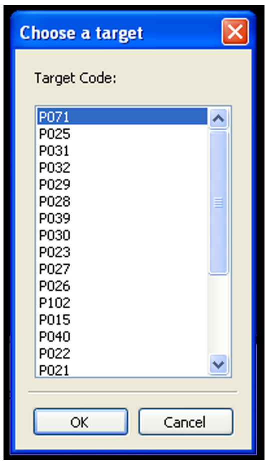


Figure 5.9: Example of the Bresenham line algorithm

5.4 PhotoModeler Interface

Now that we have extracted all the coded and 'dummy' targets from the all the images we can start loading them into PhotoModeler so that they can be processed and used. In a real world situation we would have several hundred images and it would be unpractical to simply load them all into PhotoModeler. For this reason, we ask the user to specify a certain target in the area on which he or she would like to focus. A list of all the targets in the images is displayed as in Figure 5.10. After the user has selected a target, all the images that contain that target are compared to each other find the correlation value. The correlation value is simply the number of coded targets the images have in common. Once all the relevant images are ranked, the user has to choose how many stereo pairs must be loaded into PhotoModeler®(see Figure 5.11).



P071
P025
P031
P032
P029
P028
P039
P030
P023
P027
P026
P102
P015
P040
P022
P021

Figure 5.10: Menu of Targets in images

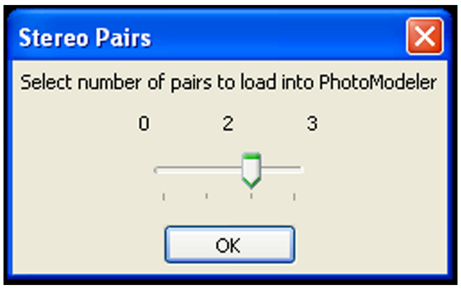


Figure 5.11: Choose number of Stereo Pairs

By using the Dynamic Data Exchange (DDE), first introduced in 1987 with the release of Windows 2.0, we can control PhotoModeler from within Python and feed our data to it. In our case PhotoModeler will act as a DDE server and we use Python to open a DDE client (a client initiates a DDE connection to a DDE server and sends commands to it).

PhotoModeler's DDE commands are ASCII text strings and take a very simple form. Only three commands were implemented - "DDE Initiate", "DDE Request" and "DDE Terminate". When both PhotoModeler and our application are running, the "DDE Initiate" command is used to establish a channel with the server name: "PhotoModeler" and the topic: "Data". Once the link is open, we can ask PhotoModeler to perform tasks by using the "DDE Request" command. When all the images and data are loaded, we close the channel by using the "DDE Terminate" command. An example of PhotoModeler®is given in Figure 5.12, where the points were loaded in by the DDD interface. Only the centerlines of the pipes are modeled and then the 3D model is exported to Microstation, where equipment and pipe specifications (diameter, material, etc.) are added.

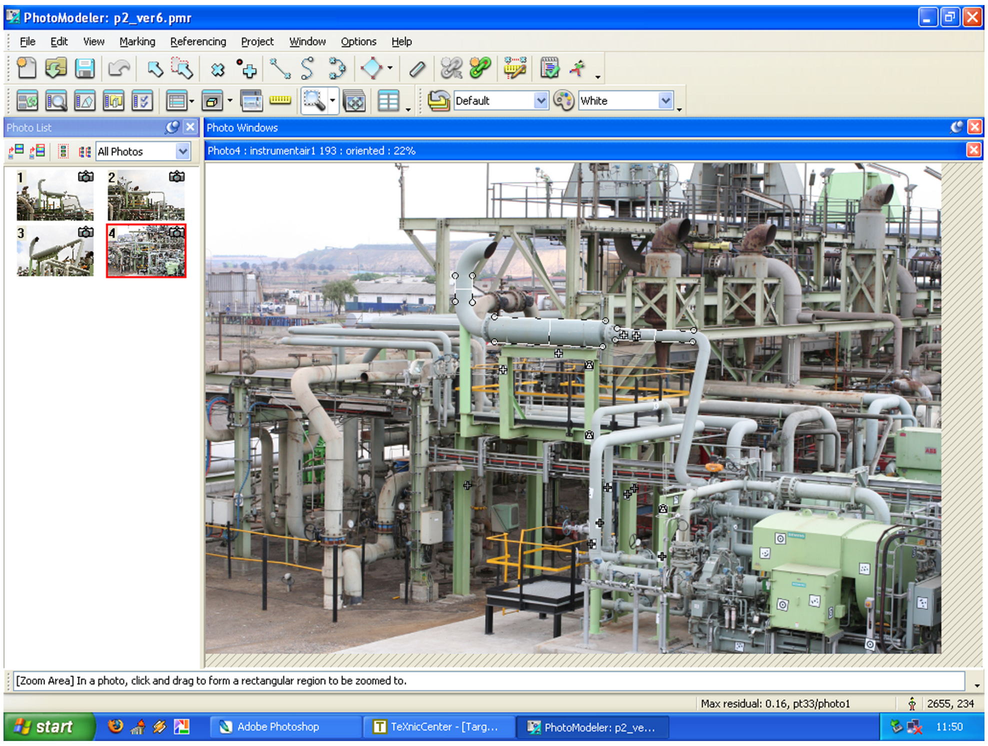

PhotoModeler: p2_ver6.pmr
File Edit View Marking Referencing Project Window Options Help
Default
White
Photo List
All Photos
Photo Windows
Photo4 : instrumentair1 193 : oriented : 22%
[Zoom Area] In a photo, click and drag to form a rectangular region to be zoomed to.
Max residual: 0.16, pt33/photo1
2655, 234
start
Adobe Photoshop
TeXnicCenter - [Targ...
PhotoModeler: p2_ve...
11:50

Figure 5.12: PhotoModeler

Chapter 6

Conclusions

The goal of this thesis was to asses wheter a photogrammetry system would be practical and effective in modeling plants in the petrochemical industry. Such a system would provide an affordable and convenient way to generate a 3D model of an existing plant. An additional application of this system would be monitoring the construction of a plant, enableing the early detection of construction errors.

The reconstruction process starts with camera calibration, after which photographs of the plant are taken using the calibrated camera. Next, one needs to identify the targets and sort the photographs into stereo pairs. The last two steps are to select a feature (i.e. pipe) and reconstruct its 3D coordinates. Each of these steps were described en this thesis, and illustrated by the reconstruction of a simple Lego object.

By implementing this simple 3D reconstruction system we were able to get useful results, suggesting that a full implementation would be effective in a practical environment. Indeed, when we investigated the system used by Bhukka (Pty) Ltd, it proved an efficient solution to the problem of digital 3D modeling, but the manual extraction of surveyed target was very time consuming.

The automatic extraction of targets from images was succesfully added to the functionality of Photomodeler®. This provides a usable photogrammetry system to reconstruct a petrochemical plant: The software of this thesis uses image processing to automatically extract and identify the surveyed targets in the photographs, thus enabling automatic stereo pair finding. The target and stereo pair information is then passed to Photomodeler®, which can then be used to do a reconstruction of any selected feature.

Recommendations

If the simple implementation of 3D reconstruction in this thesis is to be expanded to be a full working system without the use of Photomodeler®, certain areas would need to be enhanced or added. The planar calibration algorithm needs to include radial distortion modeling [39] and an iterative minimisation method, such as Levenberg-Marquardt technique.

While the normalised 8-point algorithm and essential matrix method give good results for 3D reconstruction, an implementation of the Gold Standard algorithm [15, Section 11.4.1] would improve the accuracy of the results. Further improvements can be made by using the optimal method of error minimisation, given by Hartley and Sturm [14]. To model an actual petrochemical plant, the reconstruction of objects other than points, such as lines, cylinders and boxes, need to be part of the application.

More functionality and stability can still be added to the target extraction algortihm. The algorithm is known to fail when the contrast between the darkest shadow and brightest part of the image is too great. It may be possible to segment the image and run the algorithm on seperate pieces with an adaptive threshold, but further investigation is needed.

Appendices

Appendix A

Projective Geometry

A.1 Projective Space

Two lines will almost always meet in a point, that is, unless they are parallel. A common linguistic device for getting around this is to say that the lines meet "at infinity", but infinity does not really exist and is only a convenient fiction. This is one major shortcoming in Euclidean geometry. By enhancing the Euclidean plane with points at infinity, called *ideal points*, where parallel lines will meet, we are able to resolve this difficulty.

With the addition of these ideal points Euclidean space is transformed to a new type of geometric object, projective space. We are familiar with concepts such as lines, distances, angles and points in Euclidean space, so there is nothing very mysterious about projective space - it is just an extension of Euclidean space in which two lines *always* meet in a point. The Euclidean space $\mathbb{R}^n$ can be extended to a projective space $\mathbb{P}^n$ by representing points as homogeneous vectors, which we explain in the next section.

A.2 Homogeneous Coordinates

Homogeneous coordinates have a natural application in Computer Vision as they form the basis of projective geometry. The graphical use of homogeneous coordinates is usually attributed to Roberts [25], while a good summary was first presented by Ahuja [2].

A point in $\mathbb{R}^2$ is represented by an ordered pair of real numbers, (x, y). If we add an extra coordinate, giving the triple (x, y, w) we can see that this provides us with a scale invariant representation of (x, y), with $x \leftarrow x/w$, $y \leftarrow y/w$ and $w \neq 0$. In other words (kx, ky, k) represents the same point for all values of k and are called *homogeneous coordinates* of the point (x, y).

There is however the question of what happens when w is 0. If we try to divide by the last coordinate, we get the point $(x/0, y/0)$ which is infinite. This is how points at infinity arise. Thus, by extending points to homogeneous coordinates, in effect adding points at infinity, we extend the Euclidean space to projective space.

A.3 Absolute Conic

Affine geometry is a specialisation of projective geometry in which a particular line (or plane - according to the dimension) is singled out and called the line at infinity. If we go further by specifying a single feature of that line or plane, affine geometry becomes Euclidean geometry.

If we now consider a circle in two-dimensions. There is no distinction between a circle and an ellipse in affine geometry, since arbitrary stretching of the plane preserves the line at infinity, but stretches a circle into an ellipse. In Euclidean geometry however, there is a distinct separation. Two ellipses will generally intersect in four point - a ellipse is describe by a second-degree equation, thus four solutions. Two circles, however, can not intersect in more than two points. Algebraically, we are still solving two second-degree curves, thus we should expect four solutions. What is special about circles that they only intersect in two points?

The answer is that the two circles meet in two other points as well, the *complex* points. The equation for a circle in homogeneous coordinates is

$$(x - aw)^2 + (y - bw)^2 = r^2 w^2$$

with the circles center as $(x_0, y_0, w_0)^T = (a, b, 1)^T$. We can see the complex solution of this equation, the two point that lie at infinity, are $(x_0, y_0, w_0)^T = (1, \pm i, 0)^T$. These two points are called the *circular points* and lie in the intersection of any two circles in Euclidean geometry. We can use these points to define Euclidean geometry from projective geometry. First we single out a line to be the line at infinity, and then select two point on this line to be the circular points. We may now define a circle as any conic (a curve defined by a second-degree equation) that passes through the two circular points.

The same idea may be applied to 3D geometry. If we look at the way two spheres intersect, algebra would lead us to expect a general fourth-degree curve, but instead we see they intersect in a circle. With the same line of thought as for 2D geometry we see that in homogeneous coordinates all spheres intersect the plane at infinity in a curve with the equations $x^2 + y^2 + z^2 = 0; t = 0$. This second-degree curve lying on the plane at infinity is called the *absolute conic* and is closely linked to camera calibration, as seen in Chapter 3.

Appendix B

Least-squares solution of homogeneous equations

A frequent problem in the field of reconstruction is that of solving a set of equations of the form $A\boldsymbol{x} = \mathbf{0}$. We consider an over-determined set of equations, with noise, where there are more equations than unknowns and no exact solution. The obvious solution $\boldsymbol{x} = \mathbf{0}$ is not of interest as we seek a non-zero solution. Suppose $\boldsymbol{x}$ is a solution to the set of equations, then $k\boldsymbol{x}$, for any scalar k, is also a solution. So the solution can only be determined up to a scale and to find this scale we need a constraint on the system. A reasonable constraint would be the norm of vector, $||\boldsymbol{x}|| = 1$. The problem can now be stated as

Find the $\boldsymbol{x}$ that minimises $||A\boldsymbol{x}||$ subject to $||\boldsymbol{x}|| = 1$.

Now, let $A = UDV^T$, also known as the Singular Value Decomposition (SVD) of A. The problem then requires us to minimises $||UDV^T\boldsymbol{x}||$. However, $||UDV^T\boldsymbol{x}|| = ||DV^T\boldsymbol{x}||$ and $||\boldsymbol{x}|| = ||V^T\boldsymbol{x}||$. If we write $\boldsymbol{y} = V^T\boldsymbol{x}$ the problem is now: minimise $||D\boldsymbol{y}||$ subject to $||\boldsymbol{y}|| = 1$. The solution is thus

$\boldsymbol{y} = (0, 0, ..., 0, 1)^T$ or at least, $\boldsymbol{y}$ having one non-zero entry, a 1, in the last position. Finally, $\boldsymbol{x} = V\boldsymbol{y}$ is simply the last column of V. A brief summary of the algorithm:

Objective

Given a matrix A with at least as many rows as columns, find $\boldsymbol{x}$ that minimises $\|A\boldsymbol{x}\|$ subject to $\|\boldsymbol{x}\| = 1$.

Solution

$\boldsymbol{x}$ is the last column of V, where $A = UDV^T$ is the SVD of A.

Bibliography

[1] S. Ahn, W. Rauh, and S. Kim. Circular coded target for automation of optical 3d-measurement and camera calibration. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(6):905–919, 2001.

[2] D. V. Ahuja and S. A. Coons. Geometry for construction and display. In *Readings in Computer Vision: Issues, Problems, Principles and Paradigms*, pages 652–660, San Francisco, CA, USA, 1987. Morgan Kaufmann Publishers Inc.

[3] S. Bougnoux. From projective to euclidian space under any practical situation, a criticism of self-calibration. *6th International Conference on Computer Vision*, pages 790–796, 1998.

[4] J. Bresenham. Algorithm for computer control of a digital plotter. *IBM Systems Journal*, 4(1):25–30, 1965.

[5] D. C. Brown. Close-range camera calibration. *Photogrammetric Engineering*, 37:855–866, 1971.

[6] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

[7] O. D. Faugeras. What can be seen in three dimensions with an uncalibrated stereo rig. In *ECCV '92: Proceedings of the Second European Conference on Computer Vision*, pages 563–578, London, UK, 1992. Springer-Verlag.

[8] O. D. Faugeras. *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.

[9] K. D. Gremban, C. E. Thorpe, and T. Kanade. Geometric camera calibration using systems of linear equations. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 562–567, 1988.

[10] W. I. Grosky and L. A. Tamburino. A unified approach to the linear camera calibration problem. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 663–671, 1990.

[11] R. I. Hartley. Estimation of relative camera positions for uncalibrated cameras. In *Proceedings of the European Conference on Geometric Vision*, LNCS 588, pages 579–587. Springer-Verlag, 1992.

[12] R. I. Hartley. In defense of the eight-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, 1997.

[13] R. I. Hartley, R. Gupta, and T. Chang. Stereo from uncalibrated cameras. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, pages 761–764, 1992.

[14] R. I. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68:146–157, 1997.

[15] R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2003.

[16] D. Liebowitz and A. Zisserman. Metric rectification for perspective images of planes. In *Computer Society Conference on Computer Vision and Pattern Recognition*, pages 482–488, 1998.

[17] H. C. Longuet-Higgins. A computer algorithm for reconstructing a scene from two projections. *Nature*, 293:133–135, 1981.

[18] Q. T. Luong. *Fundamental Matrix and self-calibration*. PhD thesis, University of Paris, 1992.

[19] Q. T. Luong and O. D. Faugeras. Self-calibration of a moving camera from point correspondences and fundamental matrices. *International Journal of Computer Vision*, 22(3):261–289, 1997.

[20] Q. T. Luong and T. Viéville. Canonical representations for the geometries of multiple projective views. *Computer Vision and Image Understanding*, 64(2):193–229, 1996.

[21] J. Maybank and R. Y. Tsai. A theory of self-calibration of a moving camera. *International Journal of Computer Vision*, 8(2):123–151, 1992.

[22] X. Meng and Z. Hu. A new easy camera calibration technique based on circular points. In *Pattern Recognition*, volume 36, pages 1155–1164, 2003.

[23] J. J. Morê. *Levenberg-Marquardt Algorithm: Implementation and Theory*, volume 630/1978 of *Lecture Notes in Mathematics*, pages 105–116. Springer Berlin/Heidelberg, 1978.

[24] D. V. Papadimitriou and T. J. Dennis. Epipolar line estimation and rectification for stereo image pairs. *IEEE Transactions on Image Processing*, 5(4):672–676, 1996.

[25] L. G. Roberts. *Homogeneous matrix representation and manipulation of N-dimensional constructs*. S. I. : s.n., 1 edition, 1965.

[26] P. D. Sampson. Fitting conic sections to very scattered data: An iterative refinement of the bookstein algorithm. *Computer Graphics and Image Processing*, 18:97–108, 1982.

[27] C. Schmid and A. Zisserman. Automatic line matching across views. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 00:666, 1997.

[28] R. R. Schultz and R. L. Stevenson. Extraction of high-resolution frames from video sequences. *IEEE Transactions on Image Processing*, 5(6):996–1011, 1996.

[29] V. Sequeira *et al*. 3d verification of plant design. In *Proceedings* 25th *ESARDA Symposium on Safeguards and Nuclear Material Management*, 2003.

[30] B. Triggs. Autocalibration from planar scenes. In *Computer Vision - ECCV'98*, volume 1406/1998 of *Lecture Notes in Computer Science*, page 89. Springer Berlin/Heidelberg, 1998.

[31] R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses. *IEEE Journal of Robotics and Automation*, 3(4):323–344, 1987.

[32] G. Q. Wei and S. D. Ma. Implicit and explicit camera calibration: Theory and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(5):469–480, 1994.

[33] C. Wheatstone. On some remarkable, and hitherto unobserved, phenomena of binocular vision. *Philosophical Transactions of the Royal Society of London*, 128:371–394, 1838.

[34] Wikipedia.com. 3d scanner — wikipedia, the free encyclopedia, 2007.

[35] Wikipedia.com. Camcorder — wikipedia, the free encyclopedia, 2007.

[36] Wikipedia.com. Digital camera — wikipedia, the free encyclopedia, 2007.

[37] G. Xu and Z. Zhang. *Epipolar Geometry in Stereo, Motion and Object Recognition: A Unified Approach*. Springer, 1 edition, 1996.

[38] Z. Zhang. Determining the epipolar geometry and its uncertainty: A review. *International Journal of Computer Vision*, 27(2):1573–1405, 1998.

[39] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.