3D Reconstruction and Camera Calibration from 2D Images

by

Arne Henrichsen

Submitted to the Department of Electrical Engineering
in fulfilment of the requirements for the degree of

Master of Science in Engineering

at the

UNIVERSITY OF CAPE TOWN

December 2000

Declaration

I, Arne Henrichsen, declare that this dissertation is my own work, except where otherwise stated. It is being submitted for the degree of Master of Science in Engineering at the University of Cape Town and it has not been submitted before for any degree or examination, at any other university.

Signature of Author . . . .

Cape Town
December 2000

Acknowledgements

I would like to thank my supervisor, Professor Gerhard de Jager, for the opportunity he gave me to work in the Digital Image Processing group and for his enthusiasm and guidance throughout this project.

I also wish to thank:

- The National Research Foundation and DebTech for their generous financial support.
- All the members of the Digital Image Processing group for providing an interesting and stimulating working environment, and in particular Fred Nicolls for his help with my many questions.
- Marthinus C. Briers from the Department of Geomatics at UCT for his help in calibrating my camera.
- David Liebowitz from the Department of Engineering Science at Oxford for helping me understand his calibration method.

Abstract

A 3D reconstruction technique from stereo images is presented that needs minimal intervention from the user.

The reconstruction problem consists of three steps, each of which is equivalent to the estimation of a specific geometry group. The first step is the estimation of the epipolar geometry that exists between the stereo image pair, a process involving feature matching in both images. The second step estimates the affine geometry, a process of finding a special plane in projective space by means of vanishing points. Camera calibration forms part of the third step in obtaining the metric geometry, from which it is possible to obtain a 3D model of the scene.

The advantage of this system is that the stereo images do not need to be calibrated in order to obtain a reconstruction. Results for both the camera calibration and reconstruction are presented to verify that it is possible to obtain a 3D model directly from features in the images.

Contents

List of Figures

List of Tables

List of Symbols

$\mathcal{P}^n$	—	Projective Space (*n*-dimensions)
$\mathcal{A}^n$	—	Affine Space (*n*-dimensions)
$\boldsymbol{l}$	—	Line in Projective Space
$\boldsymbol{\pi}$	—	Plane in Projective Space
$\boldsymbol{l}_\infty$	—	Line at Infinity
$\boldsymbol{\pi}_\infty$	—	Plane at Infinity
$\tilde{\boldsymbol{m}}$	—	Homogeneous Coordinate Vector of Vector $\boldsymbol{m}$
$\boldsymbol{\Omega}$	—	Absolute Conic
$\boldsymbol{\Omega}^*$	—	Dual Absolute Conic
$\boldsymbol{\omega}_\infty$	—	Image of the Absolute Conic
$\boldsymbol{\omega}^*_\infty$	—	Image of the Dual Absolute Conic
$\boldsymbol{K}$	—	Camera Calibration Matrix
$\boldsymbol{R}$	—	Rotation Matrix
$\boldsymbol{t}$	—	Translation Vector
$\boldsymbol{F}$	—	Fundamental Matrix

Nomenclature

2D — Two-dimensional space, eg: the image plane.

3D — Three-dimensional space.

CCD — Charged Coupled Device.

det($\boldsymbol{S}$) — Determinant of the square matrix $\boldsymbol{S}$.

SVD — Singular Value Decomposition of a Matrix.

LMedS — Least-Median-of-Squares Method.

SSD — Sum of Squared Differences.

MLE — Maximum Likelihood Estimate.

Chapter 1

Introduction

1.1 General Overview

The objective of this thesis is to present an automatic 3D reconstruction technique that uses only stereo images of a scene.

The topic of obtaining 3D models from images is a fairly new research field in computer vision. In photogrammetry, on the other hand, this field is well established and has been around since nearly the same time as the discovery of photography itself [20]. Whereas photogrammetrists are usually interested in building detailed and accurate 3D models from images, in the field of computer vision work is being done on automating the reconstruction problem and implementing an intelligent human-like system that is capable of extracting relevant information from image data.

This thesis presents a basic framework for doing exactly that. Only stereo image pairs are considered, as much relevant information is available on this topic.

The two images can be acquired by either two cameras at the same time or by one camera at a different time instant. It would be possible to extend the principle in this thesis to include a whole image sequence.

1.2 The 3D reconstruction problem

Structure from uncalibrated images only leads to a projective reconstruction. Faugeras [7] defines a matrix called the fundamental matrix, which describes the projective structure of stereo images. Many algorithms for determining the fundamental matrix have since been developed: a review of most of them can be found in a paper by Zhang [52]. Robust methods for determining the fundamental matrix are especially important when dealing with real image data. This image data is usually in the form of corners (high curvature points), as they can be easily represented and manipulated in projective geometry. There are various corner detection algorithms. The ones employed in this thesis are by Kitchen and Rosenfeld [23] and Harris and Stephens [16]. Alternatively, Taylor and Kriegman [46] develope a reconstruction algorithm using line segments instead of corners.

Image matching forms a fundamental part of epipolar analysis. Corners are estimated in both images independently, and the matching algorithm needs to pair up the corner points correctly. Initial matches are obtained by correlation and relaxation techniques. A new approach by Pilu [36] sets up a correlation-weighted proximity matrix and uses singular value decomposition to match up the points. A matching algorithm by Zhang et. al. [54] uses a very robust technique called *LMedS* (Least-Median-of-Squares Method), which is able to discard outliers in the list of initial matches and calculates the optimal fundamental matrix at the same time.

In order to upgrade the projective reconstruction to a metric or Euclidean one, 3D vision is divided or stratified into four geometry groups, of which projective geometry forms the basis. The four geometry strata are projective, affine, metric and Euclidean geometry. Stratification of 3D vision makes it easier to perform a reconstruction. Faugeras [9] gives an extensive background on how to achieve a reconstruction by upgrading projective to affine geometry and affine to metric and Euclidean geometry.

Affine geometry is established by finding the *plane at infinity* in projective space for both images. The usual method of finding the plane is by determining vanishing points in both images and then projecting them into space to obtain *points at infinity*. Vanishing points are the intersections of two or more imaged parallel lines. This process is unfortunately very difficult to automate, as the user generally has to select the parallel lines in the images. Some automatic algorithms try to find dominant line orientations in histograms [28]. Pollefeys [37] introduced the *modulus constraint*, from which it is possible to obtain an accurate estimation of the plane at infinity by determining infinite homographies between views. At least three views need to be present in order for the algorithm to work properly.

Camera calibration allows for an upgrade to metric geometry. Various techniques exist to recover the internal parameters of the camera involved in the imaging process. These parameters incorporate the focal length, the principal point and pixel skew. The classical calibration technique involves placing a calibration grid in the scene. The 3D coordinates of markers on the grid are known and the relationship between these and the corresponding image coordinates of the same markers allow for the camera to be calibrated. The calibration grid can be replaced by a virtual object lying in the plane at infinity, called the *absolute conic*. Various methods exist to calculate the absolute conic, and *Kruppa's equations* [19, 30, 49] form the basis of the most famous one. These equations provide constraints on the absolute conic and can be solved by knowing the fundamental matrix between at least three views. Vanishing points can also be used to calibrate a camera, as a paper by Caprile and Torre [5] shows. This idea is also used in a method by Liebowitz et. al. [27, 28, 29], which makes use of only a single view to obtain the camera calibration parameters. A new calibration technique which places a planar pattern of known dimensions in the scene, but for which 3D coordinates of markers are not known, has been developed by Zhang [51, 53]. The homography between the plane in the scene and the image plane is calculated, from which a calibration is possible.

Euclidean geometry is simply metric geometry, but incorporates the correct scale of the scene. The scale can be fixed by knowing the dimensions of a certain object in the scene.

Up to this point it is possible to obtain the 3D geometry of the scene, but as only a restricted number of features are extracted, it is not possible to obtain a very complete textured 3D model. Dense stereo matching techniques can be employed once the the camera projection matrices for both images are known. Most dense stereo algorithms operate on rectified stereo image pairs in order to reduce the search space to one dimension. Pollefeys, Koch and van Gool [39] reparameterise the images with polar coordinates, but need to employ oriented projective geometry [26] to orient the epipolar lines. Another rectification algorithm by Roy, Meunier and Cox [42] rectifies on a cylinder instead of a plane. This method is very difficult to implement as all operations are performed in 3D space. A very simple method implemented in this thesis is by Fusiello, Trucco and Verri [13, 14], which rectifies the two images by rotating the camera projection matrices around their optical centres until the focal planes become coplanar.

Two dense stereo matching algorithms have been considered. Koch [24, 25] obtains a 3D model by extracting depth from rectified stereoscopic images by means of fitting a surface to a disparity map and performing surface segmentation. A method by Fusiello, Roberto and Trucco [12] makes use of multiple correlation windows to obtain a good approximation to the disparity, from which it is possible by means of triangulation to obtain a 3D textured model.

1.3 Outline of the thesis

Chapter 2 summarises aspects of projective geometry and also deals with stratification of 3D vision. This chapter is extremely important as it gives a theoretical framework to the reconstruction problem.

The camera model is introduced in chapter 3, with the emphasis on epipolar geometry. Epipolar geometry defines the relationship between a stereo image pair. This relationship is in the form of a matrix, called the fundamental matrix. The fundamental matrix allows for a projective reconstruction, from which it is then possible to obtain a full Euclidean 3D reconstruction.

Three techniques for the estimation of the fundamental matrix are outlined in chapter 4. One of the techniques, the Least-Median-of-Squares (LMedS) method, plays a role in the point matching algorithm, as this method is able to detect false matches and at the same time calculates a robust estimate of the fundamental matrix. A linear least-squares technique is used as an initial estimate to the LmedS method.

In chapter 5, a robust matching algorithm is outlined that incorporates two different matching techniques. The matching process makes use of correlation and relaxation techniques to find a set of initial matches. With the help of the LMedS method, which makes use of the epipolar geometry that exists between the two images, the set of initial matches are refined and false matches are discarded. Some of the limitations of the matching algorithm are described at the end of the chapter.

Chapter 6 describes four different camera calibration methods with their advantages and disadvantages. Some original calibration methods are described that make use of calibration patterns inside the view of the camera. Selfcalibration is a technique that substitutes the calibration pattern with a virtual object. This object provides constraints to calculate the camera calibration matrix. It is also possible to obtain the internal parameters of the camera from only a single view. In order to achieve that, certain measurements in the scene need to be known. The last calibration method makes use of a planar calibration grid, which is imaged from different views. The correspondence between the image planes and the planar pattern is used to establish the calibration matrix.

The complete reconstruction process is presented in chapter 7. Projective, affine and metric reconstruction processes are described. The estimation of the plane at infinity is described in detail, and certain criteria are outlined that have to be met in order to obtain an accurate estimate of the plane. This chapter also describes dense stereo matching in order to obtain a 3D textured model of the scene. The reconstruction results are also presented in this chapter. Part of a box is reconstructed, verifying that the reconstruction algorithm functions properly.

Conclusions are drawn in chapter 8.

The appendix summarises various algorithms which are used throughout the thesis:

- Two corner detection algorithms are described together with an algorithm which refines the corners to subpixel accuracy.
- A straight line finder is outlined which is used to find parallel lines in the images.
- A *maximum likelihood estimate* is presented which finds the best estimate of a vanishing point.
- The *Levenberg-Marquardt* algorithm is explained as it is used in all the nonlinear minimisation routines.
- Two methods of triangulation are presented which are used in the reconstruction problem.

For a stereo image pair, the individual steps of the reconstruction algorithm are as follows:

1. Corners are detected in each image independently.
2. A set of initial corner matches is calculated.
3. The fundamental matrix is calculated using the set of initial matches.
4. False matches are discarded and the fundamental matrix is refined.
5. Projective camera matrices are established from the fundamental matrix.
6. Vanishing points on three different planes and in three different directions are calculated from parallel lines in the images.
7. The plane at infinity is calculated from the vanishing points in both images.
8. The projective camera matrices are upgraded to affine camera matrices using the plane at infinity.
9. The camera calibration matrix (established separately to the reconstruction process) is used to upgrade the affine camera matrices to metric camera matrices.

10. Triangulation methods are used to obtain a full 3D reconstruction with the help of the metric camera matrices.
11. If needed, dense stereo matching techniques are employed to obtain a 3D texture map of the model to be reconstructed.

Stereo image pairs were obtained by a $Watec$® Camera, Model: WAT-202B(PAL) and grabbed by a $Asus$® AGP-V3800 Ultra framegrabber. If the scene or model did not contain enough features needed for the reconstruction, markers were put up at strategic places around the scene. These markers were usually made up from pieces of paper with straight, parallel lines printed on them for vanishing point estimation, or stars for the corner and matching algorithms.

Chapter 2

Stratification of 3D Vision

2.1 Introduction

Euclidean geometry describes a 3D world very well. As an example, the sides of objects have known or calculable lengths, intersecting lines determine angles between them, and lines that are parallel on a plane will never meet. But when it comes to describing the imaging process of a camera, the Euclidean geometry is not sufficient, as it is not possible to determine lengths and angles anymore, and parallel lines may intersect.

3D vision can be divided into four geometry groups or strata, of which Euclidean geometry is one. The simplest group is projective geometry, which forms the basis of all other groups. The other groups include affine geometry, metric geometry and then Euclidean geometry. These geometries are subgroups of each other, metric being a subgroup of affine geometry, and both these being subgroups of projective geometry.

Each geometry has a group of transformations associated with it, which leaves certain properties of each geometry invariant. These invariants, when recovered for a certain geometry, allow for an upgrade to the next higher-level geometry. Each of these geometries will be explained in terms of their invariants and transformations in the next few sections of this chapter.

Projective geometry allows for perspective projections, and as such models the imaging process very well. Having a model of this perspective projection, it is possible to upgrade the projective geometry later to Euclidean, via the affine and metric geometries.

Algebraic and projective geometry forms the basis of most computer vision tasks, especially in the fields of *3D reconstruction from images* and *camera selfcalibration*. Section 2.2 gives an overview of projective geometry and introduces some of the notation used throughout the text. Concepts such as points, lines, planes, conics and quadrics are described in two and three dimensions. The sections that follow describe the same structures, but in terms of affine, metric and Euclidean geometry.

A standard text covering all aspects of projective and algebraic geometry is by Semple and Kneebone [44]. Faugeras applies principles of projective geometry to 3D vision and reconstruction in his book [8]. Other good introductions to projective geometry are by Mohr and Triggs [33] and by Birchfield [3]. Stratification is described by Faugeras [9] and by Pollefeys [37].

The following sections are based entirely on the introductions to projective geometry and stratification by Faugeras [8, 9] and Pollefeys [37].

2.2 Projective Geometry

2.2.1 Homogeneous Coordinates and other Definitions

A point in projective space (n-dimensions), $\mathcal{P}^n$, is represented by a $(n+1)$-vector of coordinates $\boldsymbol{x} = [x_1, \ldots, x_{n+1}]^T$. At least one of the x_i coordinates must be nonzero. Two points represented by $(n+1)$-vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are considered equal if a nonzero scalar λ exists such that $\boldsymbol{x} = \lambda \boldsymbol{y}$. Equality between points is indicated by $\boldsymbol{x} \sim \boldsymbol{y}$. Because scaling is not important in projective geometry, the vectors described above are called *homogeneous coordinates* of a point.

Homogeneous points with $x_{n+1} = 0$ are called *points at infinity* and are related to the affine geometry described in section 2.3.

A *collineation* or linear transformation of $\mathcal{P}^n$ is defined as a mapping between projective spaces which preserves collinearity of any set of points. This mapping is represented by a $(m+1) \times (n+1)$ matrix $\boldsymbol{H}$, for a mapping from $\mathcal{P}^n \mapsto \mathcal{P}^m$. Again for a nonzero scalar λ, $\boldsymbol{H}$ and $\lambda\boldsymbol{H}$ represent the same collineation. If $\boldsymbol{H}$ is a $(n+1) \times (n+1)$ matrix, then $\boldsymbol{H}$ defines a collineation from $\mathcal{P}^n$ into itself.

A *projective basis* for $\mathcal{P}^n$ is defined as any set of $(n+2)$ points of $\mathcal{P}^n$, such that no $(n+1)$ of them are linearly dependent. The set $\boldsymbol{e}_i = [0, \ldots, 1, \ldots, 0]^T$, for $i = 1, \ldots, n+1$, where 1 is in the i^{th} position, and $\boldsymbol{e}_{n+2} = [1, 1, \ldots, 1]^T$ form the *standard projective basis*. A projective point of $\mathcal{P}^n$ represented by any of its coordinate vectors $\boldsymbol{x}$ can be described as a linear combination of any $n + 1$ points of the standard basis:

$$\boldsymbol{x} = \sum_{i=1}^{n+1} x_i \boldsymbol{e}_i. \tag{2.1}$$

Any projective basis can be transformed by a collineation into a standard projective basis: "*let $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n+2}$ be $n + 2$ coordinate vectors of points in $\mathcal{P}^n$, no $n + 1$ of which are linearly dependent, i.e., a projective basis. If $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_{n+1}, \boldsymbol{e}_{n+2}$ is the standard projective basis, then there exists a nonsingular matrix $\boldsymbol{A}$ such that $\boldsymbol{A}\boldsymbol{e}_i = \lambda_i \boldsymbol{x}_i$, $i = 1, \ldots, n+2$, where the λ_i are nonzero scalars; any two matrices with this property differ at most by a scalar factor*" [8, 9].

A collineation can also map a projective basis onto a second projective basis: "*if $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{n+2}$ and $\boldsymbol{y}_1, \ldots, \boldsymbol{y}_{n+2}$ are two sets of $n + 2$ coordinate vectors such that in either set no $n + 1$ vectors are linearly dependent, i.e., form two projective basis, then there exists a nonsingular $(n + 1) \times (n + 1)$ matrix $\boldsymbol{P}$ such that $\boldsymbol{P}\boldsymbol{x}_i = \rho_i \boldsymbol{y}_i$, $i = 1, \ldots, n + 2$, where the ρ_i are scalars, and the matrix $\boldsymbol{P}$ is uniquely determined apart from a scalar factor*" [8, 9].

The proof for both above statements can be found in [8].

2.2.2 The Projective Plane

The projective space $\mathcal{P}^2$ is known as the projective plane. A point in $\mathcal{P}^2$ is defined as a 3-vector $\boldsymbol{x} = [x_1, x_2, x_3]^T$, with $(u, v) = (x_1/x_3, x_2/x_3)$ the Euclidean position on the plane. A line is also defined as a 3-vector $\boldsymbol{l} = [l_1, l_2, l_3]^T$ and having the equation of

$$\sum_{i=1}^{3} l_i x_i = 0. \tag{2.2}$$

Then a point $\boldsymbol{x}$ is located on the line if

$$\boldsymbol{l}^T \boldsymbol{x} = 0. \tag{2.3}$$

This equation can be called the *line equation*, which means that a point $\boldsymbol{x}$ is represented by a set of lines through it, or this equation is called the *point equation*, which means that a line $\boldsymbol{l}$ is represented by a set of points. These two statements show that there is no difference between points and lines in $\mathcal{P}^2$. This is called the *principle of duality*. Any theorem or statement that is true for the projective plane can be reworded by substituting points for lines and lines for points, and the resulting statement will also be true.

The equation of the line through two points $\boldsymbol{x}$ and $\boldsymbol{y}$ is

$$\boldsymbol{l} = \boldsymbol{x} \times \boldsymbol{y}, \tag{2.4}$$

which is also sometimes calculated as follows:

$$\boldsymbol{l} = [\boldsymbol{x}]_\times \boldsymbol{y}, \tag{2.5}$$

with

$$[\boldsymbol{x}]_\times = \begin{bmatrix} 0 & x_3 & -x_2 \\ -x_3 & 0 & x_1 \\ x_2 & -x_1 & 0 \end{bmatrix} \tag{2.6}$$

being the antisymmetric matrix of coordinate vector $\boldsymbol{x}$ associated with the cross product. The intersection point of two lines is also defined by the cross product: $\boldsymbol{x} = \boldsymbol{l}_1 \times \boldsymbol{l}_2$.

All the lines passing through a specific point form the *pencil of lines*. If two lines $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ are elements of this pencil, then all the other lines can be obtained as follows:

$$\boldsymbol{l} = \lambda_1 \boldsymbol{l}_1 + \lambda_2 \boldsymbol{l}_2, \tag{2.7}$$

where λ_1 and λ_2 are scalars.

Cross-Ratio

If four points $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3$ and $\boldsymbol{x}_4$ are collinear, then they can be expressed by

$$\boldsymbol{x}_i = \boldsymbol{y} + \lambda_i \boldsymbol{z}$$

for two points $\boldsymbol{y}$ and $\boldsymbol{z}$, and no point $\boldsymbol{x}_i$ coincides with $\boldsymbol{z}$. Then the *cross-ratio* is defined as follows:

$$\{\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{x}_3, \boldsymbol{x}_4\} = \frac{\lambda_1 - \lambda_3}{\lambda_1 - \lambda_4} : \frac{\lambda_2 - \lambda_3}{\lambda_2 - \lambda_4}. \tag{2.8}$$

The cross-ratio is invariant to all collineations of projective space. A similar cross-ratio can be derived for four lines: for "*four lines $\boldsymbol{l}_1$, $\boldsymbol{l}_2$, $\boldsymbol{l}_3$ and $\boldsymbol{l}_4$ of $\mathcal{P}^2$ intersecting at a point, their cross-ratio $\{\boldsymbol{l}_1, \boldsymbol{l}_2; \boldsymbol{l}_3, \boldsymbol{l}_4\}$ is defined as the cross-ratio $\{\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{x}_3, \boldsymbol{x}_4\}$ of their four points of intersection with any line $\boldsymbol{l}$ not going through their point of intersection*" [8, 9]. See figure 2.1 for a graphical explanation.

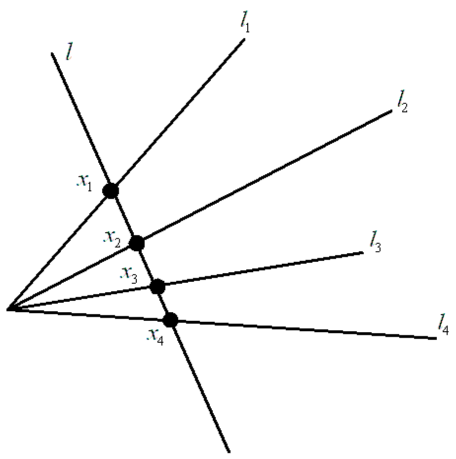


Figure 2.1: Cross-ratio of four lines:$\{\boldsymbol{l}_1, \boldsymbol{l}_2; \boldsymbol{l}_3, \boldsymbol{l}_4\}$=$\{\boldsymbol{x}_1, \boldsymbol{x}_2; \boldsymbol{x}_3, \boldsymbol{x}_4\}$. Figure obtained from [8].

Collineations

A collineation of $\mathcal{P}^2$ is defined by 3×3 invertible matrices, defined up to a scale factor. Collineations transform points, lines and pencil of lines[1] to points, lines and pencil of lines, and preserve the cross-ratios. In $\mathcal{P}^2$ collineations are called homographies and are represented by a matrix $\boldsymbol{H}$. A point $\boldsymbol{x}$ is transformed as follows:

$$\boldsymbol{x}' \sim \boldsymbol{H}\boldsymbol{x}. \tag{2.9}$$

The transformation of a line $\boldsymbol{l}$ is found by transforming the points $\boldsymbol{x}$ on the line and then finding the line defined by these points:

$$\boldsymbol{l}'^T \boldsymbol{x}' = \boldsymbol{l}^T \boldsymbol{H}^{-1} \boldsymbol{H}\boldsymbol{x} = \boldsymbol{l}^T \boldsymbol{x} = 0.$$

The transformation of the line is then as follows, with $\boldsymbol{H}^{-T} = (\boldsymbol{H}^{-1})^T = (\boldsymbol{H}^T)^{-1}$:

$$\boldsymbol{l}' \sim \boldsymbol{H}^{-T} \boldsymbol{l}. \tag{2.10}$$

Conics

In Euclidean geometry, second-order curves such as ellipses, parabolas and hyperbolas are easily defined. In projective geometry, these curves are collectively known as *conics*. A conic

[1]The *pencil of lines* is the set of lines in $\mathcal{P}^2$ passing through a fixed point.

$\boldsymbol{C}$ is defined as the locus of points of the projective plane that satisfies the following equation:

$$S(\boldsymbol{x}) = \boldsymbol{x}^T \boldsymbol{C} \boldsymbol{x} = 0$$

$$\text{or} \tag{2.11}$$

$$S(\boldsymbol{x}) = \sum_{i,j=1}^{3} c_{ij} x_i x_j = 0,$$

where $c_{ij} = c_{ji}$ which form $\boldsymbol{C}$, a 3×3 symmetric matrix defined up to a scale factor. This means that the conic depends on 5 parameters. A conic can be visualised by thinking in terms of Euclidean geometry: a circle is defined as a locus of points with constant distance from the centre, and a conic is defined as a locus of points with constant cross-ratio to four fixed points, no three of which are linearly dependent [3].

The principle of duality exists also for conics: the *dual conic* $\boldsymbol{C}^*$ or *conic envelope* is defined as the locus of all lines satisfying the following equation:

$$\boldsymbol{l}^T \boldsymbol{C}^* \boldsymbol{l} = 0, \tag{2.12}$$

where $\boldsymbol{C}^*$ is a 3×3 symmetric matrix defined up to a scale factor and depends also on 5 parameters.

Faugeras [8, 9] shows that the tangent $\boldsymbol{l}$ at a point $\boldsymbol{x}$ on a conic is defined by

$$\boldsymbol{l} = \boldsymbol{C}^T \boldsymbol{x} = \boldsymbol{C} \boldsymbol{x}. \tag{2.13}$$

Then the relationship between the conic and the dual conic is as follows: when $\boldsymbol{x}$ varies along the conic, the equation $\boldsymbol{x}^T \boldsymbol{C} \boldsymbol{x} = 0$ is satisfied and thus the tangent line $\boldsymbol{l}$ to the conic at $\boldsymbol{x}$ satisfies $\boldsymbol{l}^T \boldsymbol{C}^{-T} \boldsymbol{l} = 0$. Comparing this to equation (2.12), it shows that the tangents to a conic defined by $\boldsymbol{C}$ belong to a dual conic defined by $\boldsymbol{C}^* \sim \boldsymbol{C}^{-T}$.

Transformations of the conic and dual conic with homography $\boldsymbol{H}$ are as follows (using equations (2.9) and (2.10)):

$$\boldsymbol{x}'^T \boldsymbol{C}' \boldsymbol{x}' \sim \boldsymbol{x}^T \boldsymbol{H}^T \boldsymbol{H}^{-T} \boldsymbol{C} \boldsymbol{H}^{-1} \boldsymbol{H} \boldsymbol{x} = 0$$
$$\boldsymbol{l}'^T \boldsymbol{C}^{*'} \boldsymbol{l}' \sim \boldsymbol{l}^T \boldsymbol{H}^{-1} \boldsymbol{H} \boldsymbol{C}^* \boldsymbol{H}^T \boldsymbol{H}^{-T} \boldsymbol{l} = 0$$

and therefore

$$\boldsymbol{C}' \sim \boldsymbol{H}^{-T}\boldsymbol{C}\boldsymbol{H}^{-1} \tag{2.14}$$

$$\boldsymbol{C}^{*'} \sim \boldsymbol{H}\boldsymbol{C}^{*}\boldsymbol{H}^{T}. \tag{2.15}$$

Poles and Polars

Poles and *polars* are defined as follows: "*a point $\boldsymbol{x}$ and conic $\boldsymbol{C}$ define a line $\boldsymbol{l} = \boldsymbol{C}\boldsymbol{x}$. The line $\boldsymbol{l}$ is called the* polar *of $\boldsymbol{x}$ with respect to $\boldsymbol{C}$, and the point $\boldsymbol{x}$ is the* pole *of $\boldsymbol{l}$ with respect to $\boldsymbol{C}$. The polar line $\boldsymbol{l} = \boldsymbol{C}\boldsymbol{x}$ of the point $\boldsymbol{x}$ with respect to a conic $\boldsymbol{C}$ intersects the conic in two points at which tangent lines to $\boldsymbol{C}$ intersect at $\boldsymbol{x}$. If a point $\boldsymbol{v}_1$ lies on the polar of another point $\boldsymbol{v}_2$, then the two points are said to be conjugate with respect to the conic and satisfy $\boldsymbol{v}_1^T\boldsymbol{C}\boldsymbol{v}_2 = 0$*" [29].

Figure 2.2 shows how this is achieved.

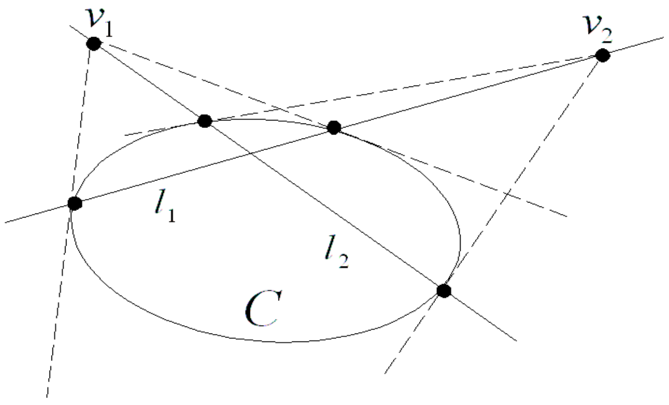


Figure 2.2: Points $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$, with polars $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$. The points $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are conjugate with respect to the conic $\boldsymbol{C}$. Figure obtained from [29].

2.2.3 The Projective Space

The space $\mathcal{P}^3$ is known as the projective space. A point of $\mathcal{P}^3$ is defined by a 4-vector $\boldsymbol{x} = [x_1, x_2, x_3, x_4]^T$. The dual entity of the point in $\mathcal{P}^3$ is a plane $\boldsymbol{\pi}$, which is also represented by a 4-vector $\boldsymbol{\pi} = [\pi_1, \pi_2, \pi_3, \pi_4]^T$ with equation of

$$\sum_{i=1}^{4} \pi_i x_i = 0. \tag{2.16}$$

A point $\boldsymbol{x}$ is located on a plane if the following equation is true:

$$\boldsymbol{\pi}^T \boldsymbol{x} = 0. \tag{2.17}$$

The structure which is analogous to the pencil of lines of $\mathcal{P}^2$ is the *pencil of planes*, the set of all planes that intersect in a certain line.

Cross-Ratio

The cross-ratio in $\mathcal{P}_3$ is defined as four planes $\boldsymbol{\pi}_1$, $\boldsymbol{\pi}_2$, $\boldsymbol{\pi}_3$ and $\boldsymbol{\pi}_4$ of $\mathcal{P}_3$ that intersect at a line $\boldsymbol{l}$. That means that the cross-ratio $\{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2; \boldsymbol{\pi}_3, \boldsymbol{\pi}_4\}$ is defined as the cross-ratio $\{\boldsymbol{l}_1, \boldsymbol{l}_2; \boldsymbol{l}_3, \boldsymbol{l}_4\}$ of their four lines of intersection with any plane $\boldsymbol{\pi}$ not going through $\boldsymbol{l}$. Another formulation is as follows: "*the cross-ratio is the cross-ratio of the four points of intersection of any line, not lying in any of the four planes, with the four planes*" [8, 9], (see figure 2.3).

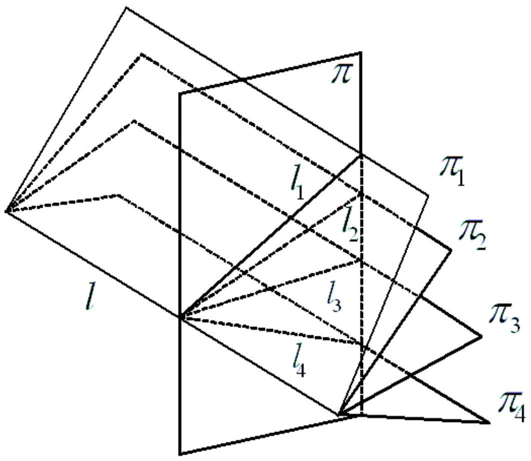


Figure 2.3: Cross-ratio of four planes:$\{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2; \boldsymbol{\pi}_3, \boldsymbol{\pi}_4\}$=$\{\boldsymbol{l}_1, \boldsymbol{l}_2; \boldsymbol{l}_3, \boldsymbol{l}_4\}$. Figure obtained from [8].

Collineations

Collineations in $\mathcal{P}^3$ are defined by 4×4 invertible matrices $\boldsymbol{T}$, defined up to a scale factor. Again it can be seen that collineations transform points, lines, planes and pencil of planes[2] to points, lines, planes and pencil of planes, and preserve the cross-ratios.

[2]The *pencil of planes* is the set of all planes intersecting in a given line.

As was the case in $\mathcal{P}^2$, transformations $\boldsymbol{T}$ of points $\boldsymbol{x}$ and planes $\boldsymbol{\pi}$ in $\mathcal{P}^3$ are as follows:

$$\boldsymbol{x}' \sim \boldsymbol{T}\boldsymbol{x} \tag{2.18}$$

and

$$\boldsymbol{\pi}' \sim \boldsymbol{T}^{-T}\boldsymbol{\pi}. \tag{2.19}$$

Quadrics

The equivalent to a conic in $\mathcal{P}^3$ is a *quadric*. A quadric is the locus of all points $\boldsymbol{x}$ satisfying:

$$S(\boldsymbol{x}) = \boldsymbol{x}^T\boldsymbol{Q}\boldsymbol{x} = 0 \quad \text{or} \quad S(\boldsymbol{x}) = \sum_{i,j=1}^{4} q_{ij}x_ix_j = 0, \tag{2.20}$$

where $\boldsymbol{Q}$ is a 4×4 symmetric matrix defined up to a scale factor. A quadric depends on 9 independent parameters.

The *dual quadric* is the locus of all planes $\boldsymbol{\pi}$ satisfying:

$$\boldsymbol{\pi}^T\boldsymbol{Q}^*\boldsymbol{\pi} = 0, \tag{2.21}$$

where $\boldsymbol{Q}^*$ is a 4×4 symmetric matrix defined up to a scale factor and also depends on 9 independent parameters.

Transformations $\boldsymbol{T}$ of the quadric and dual quadric are as follows (similar to transformations of the conic as in the previous section):

$$\boldsymbol{x}'^T\boldsymbol{Q}'\boldsymbol{x}' \sim \boldsymbol{x}^T\boldsymbol{T}^T\boldsymbol{T}^{-T}\boldsymbol{Q}\boldsymbol{T}^{-1}\boldsymbol{T}\boldsymbol{x} = 0$$
$$\boldsymbol{\pi}'^T\boldsymbol{Q}^{*'}\boldsymbol{\pi}' \sim \boldsymbol{\pi}^T\boldsymbol{T}^{-1}\boldsymbol{T}\boldsymbol{Q}^*\boldsymbol{T}^T\boldsymbol{T}^{-T}\boldsymbol{\pi} = 0$$

and therefore

$$\boldsymbol{Q}' \sim \boldsymbol{T}^{-T}\boldsymbol{Q}\boldsymbol{T}^{-1} \tag{2.22}$$
$$\boldsymbol{Q}^{*'} \sim \boldsymbol{T}\boldsymbol{Q}^*\boldsymbol{T}^T. \tag{2.23}$$

The quadric can be described as a surface of $\mathcal{P}_3$.

2.2.4 Discussion

Now that a framework for projective geometry has been created, it is possible to define 3D Euclidean space as embedded in a projective space $\mathcal{P}^3$. In a similar way, the image plane of the camera is embedded in a projective space $\mathcal{P}^2$. Then a collineation exists which maps the 3D space to the image plane, $\mathcal{P}^3 \mapsto \mathcal{P}^2$, via a 3×4 matrix. This will be dealt with in detail in the next chapter.

As was outlined, the cross-ratio stays invariant to projective transformations or collineations. The relations of *incidence, collinearity* and *tangency* are also projectively invariant.

2.3 Affine Geometry

This stratum lies between the projective and metric geometries and contains more structure than the projective stratum, but less than the metric and Euclidean ones.

2.3.1 The Affine Plane

The line in the projective plane with $x_3 = 0$ is called the *line at infinity* or $\boldsymbol{l}_\infty$. It is represented by the vector $\boldsymbol{l}_\infty = [0, 0, 1]^T$.

The affine plane can be considered to be embedded in the projective plane under a correspondence of $\mathcal{A}^2 \rightarrow \mathcal{P}^2$: $\boldsymbol{X} = [X_1, X_2]^T \rightarrow [X_1, X_2, 1]^T$. There "*is a one-to-one correspondence between the affine plane and the projective plane minus the line at infinity with equation* $x_3 = 0$" [8, 9]. For a projective point $\boldsymbol{x} = [x_1, x_2, x_3]^T$ that is not on the line at infinity, the affine parameters can be calculated as $X_1 = \frac{x_1}{x_3}$ and $X_2 = \frac{x_2}{x_3}$.

To calculate any line's point at infinity, this line needs to be simply intersected with $\boldsymbol{l}_\infty$. If such a line is defined as in equation (2.2), this intersection point is at $[-l_2, l_1, 0]^T$ or $\boldsymbol{l} \times \boldsymbol{l}_\infty$. Using equation (2.2), the vector $[-l_2, l_1]^T$ gives the direction of the affine line $l_1x_1 + l_2x_2 + l_3 = 0$. The relationship of the line at infinity and the affine plane is then as follows: any point $\boldsymbol{x} = [x_1, x_2, 0]^T$ on $\boldsymbol{l}_\infty$ gives the direction in the underlying affine plane, with the direction being parallel to the vector $[x_1, x_2]^T$.

Faugeras [9] gives a simple example which shows how the affine plane is embedded in the projective plane: considering two parallel (not identical) lines in affine space, they must have the same direction parallel to the vector $[-l_2, l_1]^T$. Then considering them as projective lines of the projective plane, they must intersect at the point $[-l_2, l_1, 0]^T$ of $\boldsymbol{l}_\infty$. That shows that two distinct parallel lines intersect at a point of $\boldsymbol{l}_\infty$.

Transformations

A point $\boldsymbol{x}$ is transformed in the affine plane as follows:

$$\boldsymbol{X}' = \boldsymbol{B}\boldsymbol{X} + \boldsymbol{b}, \tag{2.24}$$

with $\boldsymbol{B}$ being a 2×2 matrix of rank 2, and $\boldsymbol{b}$ a 2×1 vector. These transformations form a group called the affine group, which is a subgroup of the projective group and which leaves the line at infinity invariant [8, 9].

In projective space $\mathcal{P}^2$ it is then possible to define a collineation that keeps l_∞ invariant. This collineation is defined by a 3×3 matrix $\boldsymbol{A}$ of rank 3:

$$\boldsymbol{H} = \begin{bmatrix} \boldsymbol{B} & \boldsymbol{b} \\ \boldsymbol{0}_2^T & 1 \end{bmatrix}.$$

2.3.2 The Affine Space

As in the previous section, the plane at infinity $\boldsymbol{\pi}_\infty$ has equation $x_4 = 0$ and the affine space can be considered to be embedded in the projective space under a correspondence of $\mathcal{A}^3 \rightarrow \mathcal{P}^3$: $\boldsymbol{X} = [X_1, X_2, X_3]^T \rightarrow [X_1, X_2, X_3, 1]^T$. "*This is the one-to-one correspondence between the affine space and the projective space minus the plane at infinity with equation of* $x_4 = 0$" [8, 9]. Then for each projective point $\boldsymbol{x} = [x_1, x_2, x_3, x_4]^T$ that is not in that plane, the affine parameters can be calculated as $X_1 = \frac{x_1}{x_4}$, $X_2 = \frac{x_2}{x_4}$ and $X_3 = \frac{x_3}{x_4}$.

As in $\mathcal{P}^2$, the following expression gives rise to the line at infinity: if $\boldsymbol{\pi}_\infty$ is the plane at infinity of $\mathcal{P}^3$ and $\boldsymbol{\pi}$ is a plane of $\mathcal{P}^3$ not equal to $\boldsymbol{\pi}_\infty$, then $\boldsymbol{\pi} \times \boldsymbol{\pi}_\infty$ is the line at infinity on $\boldsymbol{\pi}$. Therefore, each plane of equation (2.16) intersects the plane at infinity along a line that is its line at infinity.

As in $\mathcal{P}^2$, it can be seen that any point $\boldsymbol{x} = [x_1, x_2, x_3, 0]^T$ on $\boldsymbol{\pi}_\infty$ represents the direction parallel to the vector $[x_1, x_2, x_3]^T$. This means that two distinct affine parallel planes can be considered as two projective planes intersecting at a line in the plane at infinity $\boldsymbol{\pi}_\infty$.

Transformations

Affine transformations of space can be written exactly as in equation (2.24), but with $\boldsymbol{B}$ being a 3×3 matrix of rank 3, and $\boldsymbol{b}$ a 3×1 vector. Writing the affine transformation using homogeneous coordinates, this can be rewritten as in equation (2.18) with

$$\boldsymbol{T}_A \sim \begin{bmatrix} \boldsymbol{B} & \boldsymbol{b} \\ \boldsymbol{0}_3^T & 1 \end{bmatrix}. \tag{2.25}$$

To upgrade a specific projective representation to an affine representation, a transformation needs to be applied which brings the plane at infinity to its canonical position (i.e. $\boldsymbol{\pi}_\infty = [0, 0, 0, 1]^T$) [37]. Such a transformation should satisfy the following (as in equation (2.19)):

$$\begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \sim \boldsymbol{T}^{-T}\boldsymbol{\pi}_\infty \quad \text{or} \quad \boldsymbol{T}^T \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \sim \boldsymbol{\pi}_\infty \tag{2.26}$$

The above equation determines the fourth row of $\boldsymbol{T}$ and all other elements are not constrained [37]:

$$\boldsymbol{T}_{PA} \sim \begin{bmatrix} \boldsymbol{I}_{3\times4} \\ \boldsymbol{\pi}_\infty^T \end{bmatrix}, \tag{2.27}$$

where the last element of $\boldsymbol{\pi}_\infty$ is scaled to 1. The identity matrix $\boldsymbol{I}$ can be generalised by $\boldsymbol{I}_{3\times4} = [\boldsymbol{A}_{3\times3} \quad \boldsymbol{0}_3]$. Then every transformation in this form, with $\det(\boldsymbol{A}) \neq 0$, will map $\boldsymbol{\pi}_\infty$ to $[0, 0, 0, 1]^T$.

2.3.3 Discussion

The invariants of the affine stratum are clearly the points, lines and planes at infinity. These form an important aspect of camera calibration and 3D reconstruction, as will be seen in later chapters.

As is shown in the previous section, obtaining the plane at infinity in a specific projective representation allows for an upgrade to an affine representation. The plane at infinity can be calculated by finding three vanishing points in the images. This will be explained in more detail in chapter 7.

2.4 Metric Geometry

This stratum corresponds to the group of *similarities*. The transformations in this group are Euclidean transformations such as rotation and translation. The metric stratum allows for a complete reconstruction up to an unknown scale.

2.4.1 The Metric Plane

Affine transformations can be adapted to not only preserve the line at infinity, but to also preserve two points on that line called the *absolute points* or *circular points*. The circular points are two complex conjugate points lying on the line at infinity [44]. They are represented by $\boldsymbol{I} = [1, i, 0]^T$ and $\boldsymbol{J} = [1, -i, 0]^T$ with $i = \sqrt{-1}$.

Making use of equation (2.24) and imposing the constraint that $\boldsymbol{I}$ and $\boldsymbol{J}$ be invariant, the following is obtained:

$$\frac{1}{i} = \frac{b_{11}1 + b_{12}i + b_{1}0}{b_{21}1 + b_{22}i + b_{2}0}$$

$$\frac{1}{-i} = \frac{b_{11}1 - b_{12}i + b_{1}0}{b_{21}1 - b_{22}i + b_{2}0}$$

which results in

$$\begin{aligned} (b_{11} - b_{22})i - (b_{12} + b_{21}) &= 0 \\ -(b_{11} - b_{22})i - (b_{12} + b_{21}) &= 0. \end{aligned}$$

Then $b_{11} - b_{22} = b_{12} + b_{21} = 0$ and the following transformation is obtained:

$$\boldsymbol{X}' = c \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \boldsymbol{X} + \boldsymbol{b}, \tag{2.28}$$

where $c > 0$ and $0 \leq \alpha < 2\pi$. This transformation can be interpreted as follows: the affine point $\boldsymbol{X}$ is first rotated by an angle α around the origin, then scaled by c and then translated by $\boldsymbol{b}$.

Circular points have the special property in that they can be used to determine the angle between two lines. This angle is calculated by the *Laguerre formula*:

$$\alpha = \frac{1}{2i} \log(\{\boldsymbol{l}_1, \boldsymbol{l}_2; \boldsymbol{i}_m, \boldsymbol{j}_m\}). \tag{2.29}$$

Stated in words: "*the angle α between two lines $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ can be defined by considering their point of intersection m and the two lines $\boldsymbol{i}_m$ and $\boldsymbol{j}_m$ joining m to the absolute points $\boldsymbol{I}$ and $\boldsymbol{J}$*" [8, 9].

The Laguerre formula can also be stated differently: it is equal to the cross-ratio of the four points $\boldsymbol{I}$, $\boldsymbol{J}$, $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ of intersection of the four lines with the line at infinity (see figure 2.4).

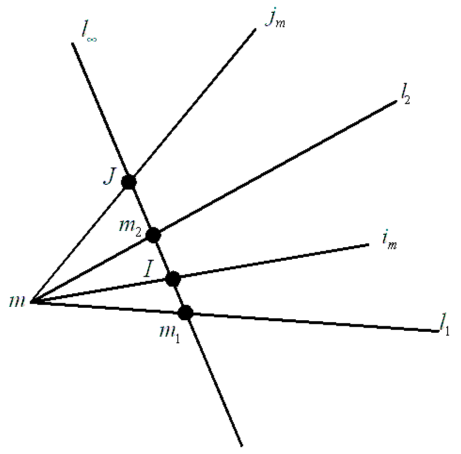


Figure 2.4: Illustration of the Laguerre formula in $\mathcal{P}^2$. Figure obtained from [8, 9].

The two lines $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$ are perpendicualar if the cross-ratio $\{\boldsymbol{l}_1, \boldsymbol{l}_2; \boldsymbol{i}_m, \boldsymbol{j}_m\}$ is equal to -1, because $e^{i\pi} = \cos\pi + i\sin\pi = -1$ [8, 9].

2.4.2 The Metric Space

In metric space, affine transformations are adapted to leave the absolute conic invariant. The absolute conic $\boldsymbol{\Omega}$ is obtained as the intersection of the quadric of equation $\sum_{i=1}^{4} x_i^2 = 0$ with $\boldsymbol{\pi}_\infty$:

$$\sum_{i=1}^{4} x_i^2 = x_4 = 0, \tag{2.30}$$

which can be interpreted as a circle of radius $i = \sqrt{-1}$, an imaginary circle in the plane at infinity [8, 9, 37]. All the points on $\boldsymbol{\Omega}$ have complex coordinates, which means that if $\boldsymbol{x}$ is a point on $\boldsymbol{\Omega}$, then the complex conjugate point $\bar{\boldsymbol{x}}$ is also on $\boldsymbol{\Omega}$.

Laguerre formula for $\mathcal{P}^3$ is as follows: "*the angle α between two planes $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$ can be defined by considering their line of intersection $\boldsymbol{l}$ and the two planes $\boldsymbol{i}_l$ and $\boldsymbol{j}_l$ going through $\boldsymbol{l}$ and tangent to the absolute conic $\boldsymbol{\Omega}$*" [8, 9]:

$$\alpha = \frac{1}{2i}\log(\{\boldsymbol{\pi}_1, \boldsymbol{\pi}_2; \boldsymbol{i}_l, \boldsymbol{j}_l\}). \tag{2.31}$$

(See figure 2.5.)

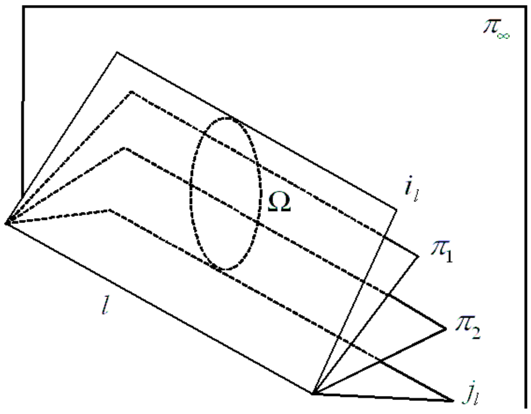


Figure 2.5: Illustration of the Laguerre formula in $\mathcal{P}^3$. Figure obtained from [8].

Affine transformations which keep $\boldsymbol{\Omega}$ invariant are written as follows:

$$\boldsymbol{X}' = c\boldsymbol{C}\boldsymbol{X} + \boldsymbol{b}, \tag{2.32}$$

where $c > 0$ and $\boldsymbol{C}$ is orthogonal: $\boldsymbol{C}\boldsymbol{C}^T = \boldsymbol{I}_{3\times 3}$. Writing the affine transformation using homogeneous coordinates, this can be rewritten as in equation (2.18) with

$$\boldsymbol{T}_M \sim \begin{bmatrix} c\boldsymbol{C} & \boldsymbol{b} \\ \boldsymbol{0}_3^T & 1 \end{bmatrix}. \tag{2.33}$$

The absolute conic $\boldsymbol{\Omega}$ is represented by two equations as in equation (2.30). The dual absolute conic $\boldsymbol{\Omega}^*$ can be represented as a single quadric [37]:

$$\boldsymbol{\Omega}^* \sim \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \tag{2.34}$$

which is its canonical form. The image of the absolute conic $\boldsymbol{\omega}_\infty$ and the image of the dual absolute conic $\boldsymbol{\omega}^*_\infty$ are the 2D representations of the conics. Their canonical forms are:

$$\boldsymbol{\omega}_\infty \sim [\boldsymbol{I}_{3\times3}] \quad \text{and} \quad \boldsymbol{\omega}^*_\infty \sim [\boldsymbol{I}_{3\times3}].$$

To upgrade the recovered affine representation of the previous section to a metric one, the absolute conic needs to be identified. This is done by an affine transformation which brings the absolute conic to its canonical position, or stated differently, from its canonical position to its actual position in the affine representation [37].

Combining equations (2.23) and (2.25), the following is obtained:

$$\boldsymbol{\Omega}^* \sim \begin{bmatrix} \boldsymbol{B} & \boldsymbol{b} \\ \boldsymbol{0}_3^T & 1 \end{bmatrix} \begin{bmatrix} \boldsymbol{I}_{3\times3} & \boldsymbol{0}_3 \\ \boldsymbol{0}_3^T & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{B}^T & \boldsymbol{0}_3 \\ \boldsymbol{b}^T & 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{B}\boldsymbol{B}^T & \boldsymbol{0}_3 \\ \boldsymbol{0}_3^T & 0 \end{bmatrix}. \tag{2.35}$$

The image of the absolute conic and the image of the dual absolute conic have then the following form:

$$\boldsymbol{\omega}_\infty = \boldsymbol{B}^{-T}\boldsymbol{B}^{-1} \quad \text{and} \quad \boldsymbol{\omega}^*_\infty = \boldsymbol{B}\boldsymbol{B}^T. \tag{2.36}$$

It is then possible to upgrade from affine to metric using the following transformation matrix:

$$\boldsymbol{T}_{AM} = \begin{bmatrix} \boldsymbol{B}^{-1} & \boldsymbol{0}_3 \\ \boldsymbol{0}_3^T & 0 \end{bmatrix}, \tag{2.37}$$

where $\boldsymbol{B}$ can be obtained via *Cholesky decomposition* [15]. As will be seen in later chapters, the matrix $\boldsymbol{B}$ is set equal to the camera calibration matrix.

2.4.3 Discussion

The absolute conic is the invariant variable of the metric stratum. Two other invariants in this group not mentioned before are *relative distances* and *angles*.

As the upgrade from an affine to a metric representation requires the camera calibration matrix, this section is closely related to the topic of camera calibration, which will be described in chapter 6.

2.5 Euclidean Geometry

Euclidean geometry is the same as metric geometry, the only difference being that the relative lengths are upgraded to absolute lengths. This means that the Euclidean transformation matrix is the same as in equation (2.33), but without the scaling factor:

$$\boldsymbol{T}_E \sim \begin{bmatrix} \boldsymbol{C} & \boldsymbol{b} \\ \boldsymbol{0}_3^T & 1 \end{bmatrix}. \tag{2.38}$$

2.6 Notations

Throughout the thesis, bold symbols represent vectors and matrices. In the following chapters, the following notation is used to represent the homogeneous coordinates of a vector: $\boldsymbol{m} = [x, y]^T \rightarrow \tilde{\boldsymbol{m}} = [\boldsymbol{m}, 1]^T$.

Chapter 3

Camera Model and Epipolar Geometry

3.1 Introduction

This chapter introduces the camera model and defines the *epipolar* or *two view* geometry.

A perspective camera model is described in section 3.2, which corresponds to the *pinhole* camera. It is assumed throughout this thesis that effects such as radial distortion are negligible and are thus ignored.

Section 3.3 defines the epipolar geometry that exists between two cameras. A special matrix will be defined that incorporates the epipolar geometry and forms the building block of the reconstruction problem.

3.2 Camera Model

A camera is usually described using the *pinhole model*. As mentioned in section 2.2, there exists a collineation which maps the projective space to the camera's retinal plane: $\mathcal{P}^3 \rightarrow \mathcal{P}^2$. Then the coordinates of a 3D point $\boldsymbol{M} = [X, Y, Z]^T$ in a Euclidean world coordinate system and the retinal image coordinates $\boldsymbol{m} = [u, v]^T$ are related by the following equation:

$$s\tilde{\boldsymbol{m}} = \boldsymbol{P}\tilde{\boldsymbol{M}}, \tag{3.1}$$

where s is a scale factor, $\tilde{\boldsymbol{m}} = [u, v, 1]^T$ and $\tilde{\boldsymbol{M}} = [X, Y, Z, 1]^T$ are the homogeneous coordinates of vector $\boldsymbol{m}$ and $\boldsymbol{M}$, and $\boldsymbol{P}$ is a 3×4 matrix representing the collineation: $\mathcal{P}^3 \rightarrow \mathcal{P}^2$. $\boldsymbol{P}$ is called the perspective projection matrix.

Figure 3.1 illustrates this process. The figure shows the case where the projection centre is placed at the origin of the world coordinate frame and the retinal plane is at $Z = f = 1$. Then $u = \frac{fX}{Z}$, $v = \frac{fY}{Z}$ and

$$\boldsymbol{P} = [\boldsymbol{I}_{3\times3} \quad \boldsymbol{0}_3]. \tag{3.2}$$

The optical axis passes through the centre of projection (camera) C and is orthogonal to the retinal plane. The point $\boldsymbol{c}$ is called the principal point, which is the intersection of the optical axis with the retinal plane. The focal length f of the camera is also shown, which is the distance between the centre of projection and the retinal plane.

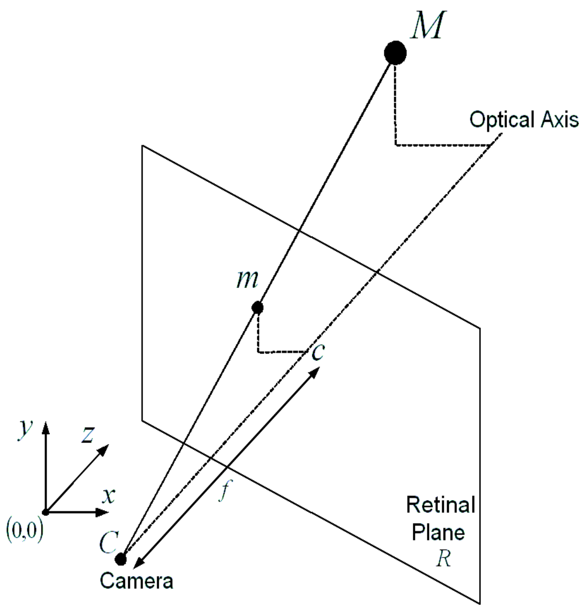


Figure 3.1: Perspective Projection.

If only the perspective projection matrix $\boldsymbol{P}$ is available, it is possible to recover the coordinates of the optical centre or camera.

The world coordinate system is usually defined as follows: the positive Y-direction is pointing upwards, the positive X-direction is pointing to the right and the positive Z-direction is pointing into the page.

3.2.1 Camera Calibration Matrix

The camera calibration matrix, denoted by $\boldsymbol{K}$, contains the intrinsic parameters of the camera used in the imaging process. This matrix is used to convert between the retinal plane and the actual image plane:

$$\boldsymbol{K} = \begin{bmatrix} \frac{f}{p_u} & (\tan\alpha)\frac{f}{p_v} & u_0 \\ 0 & \frac{f}{p_v} & v_0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{3.3}$$

Here, the focal length f acts as a scale factor. In a normal camera, the focal length mentioned above does not usually correspond to 1. It is also possible that the focal length changes during an entire imaging process, so that for each image the camera calibration matrix needs to be reestablished.

The values p_u and p_v represent the width and height of the pixels in the image, $\boldsymbol{c} = [u_0, v_0]^T$ is the principal point and α is the skew angle. This is illustrated in figure 3.2.

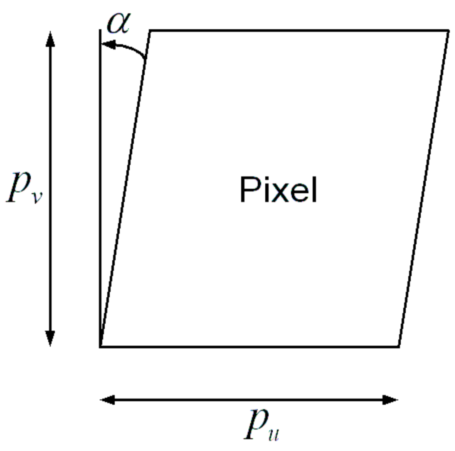


Figure 3.2: Illustration of pixel skew.

It is possible to simplify the above matrix:

$$\boldsymbol{K} = \begin{bmatrix} f_u & s & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.4}$$

where f_u and f_v are the focal lengths measured in width and height of the pixels, s represents the pixel skew and the ratio $f_u : f_v$ characterises the aspect ratio of the camera.

It is possible to use the camera calibration matrix to transform points from the retinal plane to points on the image plane:

$$\tilde{\boldsymbol{m}} = \boldsymbol{K}\tilde{\boldsymbol{m}}_{\mathcal{R}}. \tag{3.5}$$

The estimation of the camera calibration matrix is described in chapter 6.

3.2.2 Camera Motion

Motion in a 3D scene is represented by a *rotation* matrix $\boldsymbol{R}$ and a *translation* vector $\boldsymbol{t}$. The motion of the camera from coordinate $\boldsymbol{C}_1$ to $\boldsymbol{C}_2$ is then described as follows:

$$\tilde{\boldsymbol{C}}_2 = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}_3^T & 1 \end{bmatrix} \tilde{\boldsymbol{C}}_1, \tag{3.6}$$

where $\boldsymbol{R}$ is the 3×3 rotation matrix and $\boldsymbol{t}$ the translation in the X-, Y- and Z- directions. The motion of scene points is equivalent to the inverse motion of the camera (Pollefeys [37] defines this as the other way around) :

$$\tilde{\boldsymbol{M}}_2 = \begin{bmatrix} \boldsymbol{R}^T & -\boldsymbol{R}^T\boldsymbol{t} \\ \boldsymbol{0}_3^T & 1 \end{bmatrix} \tilde{\boldsymbol{M}}_1. \tag{3.7}$$

Equation (3.1) with equations (3.2), (3.5) and (3.6) then redefine the perspective projection matrix:

$$s\tilde{\boldsymbol{m}} = \boldsymbol{K}\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix}\tilde{\boldsymbol{M}}, \tag{3.8}$$

where $\boldsymbol{P} = \boldsymbol{K}\begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix}$.

3.3 Epipolar Geometry

The epipolar geometry exists between a two camera system. With reference to figure 3.3, the two cameras are represented by $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$.

Points $\boldsymbol{m}_1$ in the first image and $\boldsymbol{m}_2$ in the second image are the imaged points of the 3D point $\boldsymbol{M}$. Points $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$ are the so-called *epipoles*, and they are the intersections of the line joining the two cameras $\boldsymbol{C}_1$ and $\boldsymbol{C}_2$ with both image planes or the projection of the cameras in the

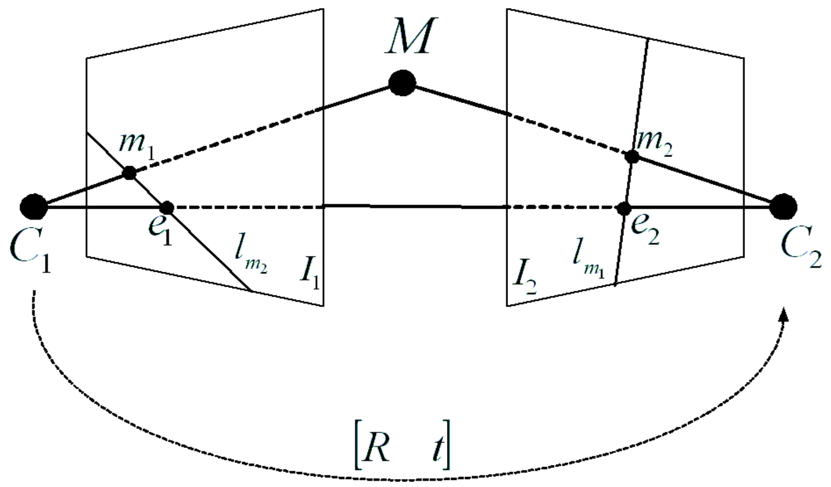


Figure 3.3: Epipolar Geometry.

opposite image. The plane formed with the three points $< \boldsymbol{C}_1\boldsymbol{M}\boldsymbol{C}_2 >$ is called the *epipolar plane*. The lines $\boldsymbol{l}_{m_1}$ and $\boldsymbol{l}_{m_2}$ are called the *epipolar lines* and are formed when the epipoles and image points are joined.

The point $\boldsymbol{m}_2$ is constrained to lie on the epipolar line $\boldsymbol{l}_{m_1}$ of point $\boldsymbol{m}_1$. This is called the *epipolar constraint*. To visualise it differently: the epipolar line $\boldsymbol{l}_{m_1}$ is the intersection of the epipolar plane mentioned above with the second image plane $\boldsymbol{I}_2$. This means that image point $\boldsymbol{m}_1$ can correspond to any 3D point (even points at infinity) on the line $< \boldsymbol{C}_1\boldsymbol{M} >$ and that the projection of $< \boldsymbol{C}_1\boldsymbol{M} >$ in the second image $\boldsymbol{I}_2$ is the line $\boldsymbol{l}_{m_1}$. All epipolar lines of the points in the first image pass through the epipole $\boldsymbol{e}_2$ and form thus a pencil of planes containing the baseline $< \boldsymbol{C}_1\boldsymbol{C}_2 >$.

The above definitions are symmetric, in a way such that the point of $\boldsymbol{m}_1$ must lie on the epipolar line $\boldsymbol{l}_{m_2}$ of point $\boldsymbol{m}_2$.

Expressing the epipolar constraint algebraically, the following equation needs to be satisfied in order for $\boldsymbol{m}_1$ and $\boldsymbol{m}_2$ to be matched:

$$\tilde{\boldsymbol{m}}_2^T \boldsymbol{F} \tilde{\boldsymbol{m}}_1 = 0, \tag{3.9}$$

where $\boldsymbol{F}$ is a 3×3 matrix called the *fundamental matrix*. The following equation also holds:

$$\boldsymbol{l}_{m_1} = \boldsymbol{F}\tilde{\boldsymbol{m}}_1, \tag{3.10}$$

since the point $\boldsymbol{m}_2$ corresponding to point $\boldsymbol{m}_1$ belongs to the line $\boldsymbol{l}_{m_1}$ [31]. The role of the images can be reversed and then:

$$\tilde{\boldsymbol{m}}_1^T \boldsymbol{F}^T \tilde{\boldsymbol{m}}_2 = 0,$$

which shows that the fundamental matrix is changed to its transpose.

Making use of equation (3.8), if the first camera coincides with the world coordinate system then

$$\begin{aligned} s_1\tilde{\boldsymbol{m}}_1 &= \boldsymbol{K}_1 \begin{bmatrix} \boldsymbol{I}_{3\times 3} & \boldsymbol{0}_3 \end{bmatrix} \tilde{\boldsymbol{M}} \\ s_2\tilde{\boldsymbol{m}}_2 &= \boldsymbol{K}_2 \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \end{bmatrix} \tilde{\boldsymbol{M}}, \end{aligned}$$

where $\boldsymbol{K}_1$ and $\boldsymbol{K}_2$ are the camera calibration matrices for each camera, and $\boldsymbol{R}$ and $\boldsymbol{t}$ describe a transformation (rotation and translation) which brings points expressed in the first coordinate system to the second one. The fundamental matrix can then be expressed as follows:

$$\boldsymbol{F} = \boldsymbol{K}_2^{-T}[\boldsymbol{t}]_x \boldsymbol{R}\boldsymbol{K}_1^{-1}, \tag{3.11}$$

where $[\boldsymbol{t}]_x$ is the antisymmetric matrix as described in equation (2.6).

Since $\det([\boldsymbol{t}]_x) = 0$, $\det(\boldsymbol{F}) = 0$ and $\boldsymbol{F}$ is of rank 2 [52]. The fundamental matrix is also only defined up to a scalar factor, and therefore it has seven degrees of freedom (7 independent parameters among the 9 elements of $\boldsymbol{F}$).

A note on the fundamental matrix: if the intrinsic parameters of the camera are known, such as in equation (3.11), then the fundamental matrix is called the *essential matrix* [31].

Another property of the fundamental matrix is derived from equations (3.9) and (3.10):

$$\boldsymbol{F}\tilde{\boldsymbol{e}}_1 = \boldsymbol{F}^T\tilde{\boldsymbol{e}}_2 = 0. \tag{3.12}$$

Clearly, the epipolar line of epipole $\boldsymbol{e}_1$ is $\boldsymbol{F}\tilde{\boldsymbol{e}}_1$.

Chapter 4

Fundamental Matrix Estimation

4.1 Introduction

The whole 3D reconstruction process relies heavily on a robust estimation of the fundamental matrix, which is able to detect outliers in the correspondences. This chapter will explain how the fundamental matrix is calculated using a robust method incorporating both a linear and nonlinear method. This chapter is based on descriptions by Zhang [50, 52] and by Luong and Faugeras [31].

As the fundamental matrix has only seven degrees of freedom, it is possible to estimate $\boldsymbol{F}$ directly using only 7 point matches. In general more than 7 point matches are available and a method for solving the fundamental matrix using 8 point matches is given in section 4.2. The points in both images are usually subject to noise and therefore a minimisation technique is implemented and described in section 4.3. A robust method is described in section 4.4 which allows for outliers in the list of matches. This is very useful as the technique will ignore these false matches in the estimation of the fundamental matrix. A short comparison with another robust method, called RANSAC, is given in section 4.5.

4.2 Linear Least-Squares Technique

Having matched a corner point $\boldsymbol{m}_{1i} = [u_{1i}, v_{1i}]^T$ in the first image with a corner point $\boldsymbol{m}_{2i} = [u_{2i}, v_{2i}]^T$ in the second image, the epipolar equation can be written as follows:

$$\tilde{\boldsymbol{m}}_{2i}^T \boldsymbol{F} \tilde{\boldsymbol{m}}_{1i} = 0. \tag{4.1}$$

This equation can be rewritten as a linear and homogeneous equation in the 9 unknown coefficients of matrix $\boldsymbol{F}$:

$$\boldsymbol{u}_i^T \boldsymbol{f} = 0, \tag{4.2}$$

where

$$\begin{aligned}\boldsymbol{u}_i &= [u_{1i}u_{2i}, v_{1i}u_{2i}, u_{2i}, u_{1i}v_{2i}, v_{1i}v_{2i}, v_{2i}, u_{1i}, v_{1i}, 1]^T \\ \boldsymbol{f} &= [F_{11}, F_{12}, F_{13}, F_{21}, F_{22}, F_{23}, F_{31}, F_{32}, F_{33}]^T\end{aligned}$$

and F_{ij} is the element of $\boldsymbol{F}$ at row i and column j. If n corner point matches are present and by stacking equation (4.2), the following linear system is obtained:

$$\boldsymbol{U}_n \boldsymbol{f} = 0,$$

where

$$\boldsymbol{U}_n = [\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n]^T.$$

If 8 or more corner point correspondences are present and ignoring the rank-2 constraint, a least-squares method can be used to solve

$$\min_{\boldsymbol{F}} \sum_i (\tilde{\boldsymbol{m}}_{2i}^T \boldsymbol{F} \tilde{\boldsymbol{m}}_{1i})^2, \tag{4.3}$$

which can be rewritten as:

$$\min_{\boldsymbol{f}} \|\boldsymbol{U}_n \boldsymbol{f}\|^2.$$

Various methods exists to solve for $\boldsymbol{f}$. They are called the 8-*point algorithms*, as 8 or more points are needed to solve for $\boldsymbol{f}$. One of the methods sets one of the coefficients of $\boldsymbol{F}$ to 1 and then solves equation (4.3) using a linear least-squares technique [52].

A second method imposes a constraint on the norm of $\boldsymbol{f}$ (i.e. $\|\boldsymbol{f}\| = 1$), and the above linear system can be solved using Eigen analysis [31, 52, 54]. The solution will then be the unit eigenvector of matrix $\boldsymbol{U}_n^T \boldsymbol{U}_n$ associated with the smallest eigenvalue, and can be found via *Singular Value Decomposition* [15].

The problem with this computation is that it is very sensitive to noise, even when a large number of matches are present. A reason for this is that the *rank-2* constraint of the fundamental matrix (i.e. $\det(\boldsymbol{F}) = 0$) is not satisfied [52, 54].

Hartley [18] challenges the view that the 8-point algorithms are very noisy in calculations and shows that by normalising the coordinates of the matched points, better results are obtained. One of the normalisation techniques he is using is non-isotropic scaling, where the centroid of the points is translated to the origin. After the translation the points form a cloud about the origin, which is scaled such that it appears to be symmetric circular with a radius of one (the two principal moments of the set of points are equal to one). The steps of the translation and scaling are as follows: for all points $\tilde{\boldsymbol{m}}_i$, $(i, \ldots, N)$, matrix $\sum_i \tilde{\boldsymbol{m}}_i \tilde{\boldsymbol{m}}_i^T$ is formed. As this matrix is symmetric and positive definite, *Cholesky decomposition* [15] will result in:

$$\sum_{i=1}^{N} \tilde{\boldsymbol{m}}_i \tilde{\boldsymbol{m}}_i^T = N\boldsymbol{A}\boldsymbol{A}^T,$$

where matrix $\boldsymbol{A}$ is upper triangular. The above equation can be rewritten:

$$\sum_{i=1}^{N} \boldsymbol{A}^{-1} \tilde{\boldsymbol{m}}_i \tilde{\boldsymbol{m}}_i^T \boldsymbol{A}^{-T} = N\boldsymbol{I},$$

where $\boldsymbol{I}$ is the identity matrix. Setting $\tilde{\boldsymbol{m}}'_i = \boldsymbol{A}^{-1}\tilde{\boldsymbol{m}}_i$, the equation for the transformed points becomes:

$$\sum_{i=1}^{N} \tilde{\boldsymbol{m}}'_i \tilde{\boldsymbol{m}}'^T_i = N\boldsymbol{I}.$$

This shows that the transformed points have their centroid at the origin and the two principal moments are both equal to one. The above transformation is applied to points in both images, yielding two transformation matrices $\boldsymbol{A}_1$ and $\boldsymbol{A}_2$.

After estimating the fundamental matrix $\boldsymbol{F}'$ corresponding to the normalised point coordinates using the 8-point algorithm described above, the fundamental matrix $\boldsymbol{F}$ corresponding to the original unnormalised point coordinates is calculated as follows:

$$\boldsymbol{F} = \boldsymbol{A}_2^T \boldsymbol{F}' \boldsymbol{A}_1.$$

4.3 Minimising the Distances to Epipolar Lines

To satisfy the *rank-2* constraint of the fundamental matrix, $\boldsymbol{F}$ can be written in terms of 7 parameters [52, 50]. Therefore $\boldsymbol{F}$ can be parameterised as follows:

$$\begin{bmatrix} a & b & -ax_1 - by_1 \\ c & d & -cx_1 - dy_1 \\ -ax_2 - cy_2 & -bx_2 - dy_2 & (ax_1 + by_1)x_2 + (cx_1 + dy_1)y_2 \end{bmatrix}. \tag{4.4}$$

The parameters (x_1, y_1) and (x_2, y_2) are the coordinates of the two epipoles $\boldsymbol{e}_1$ and $\boldsymbol{e}_2$. The four parameters (a, b, c, d) define the relationship between the orientations of the two pencils of epipolar lines [50]. The matrix is normalised by dividing the four parameters (a, b, c, d) by the largest in absolute value.

The fundamental matrix in the previous section is used as an initial guess and to estimate the two epipoles. The following technique is used by Zhang [50] to calculate the two epipoles. If

$$\boldsymbol{M} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T$$

is the *Singular Value Decomposition* of a matrix $\boldsymbol{M}$ [15], then

$$\boldsymbol{D} = \begin{bmatrix} d_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & d_3 \end{bmatrix}$$

is the diagonal matrix satisfying $d_1 \geq d_2 \geq d_3 \geq 0$, where d_i is the i^{th} singular value, and $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices. Then

$$\boldsymbol{F} = \boldsymbol{U}\hat{\boldsymbol{D}}\boldsymbol{V}^T, \tag{4.5}$$

where

$$\hat{\boldsymbol{D}} = \begin{bmatrix} ds_1 & 0 & 0 \\ 0 & d_2 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

satisfies the *rank-2* constraint of the fundamental matrix. The epipoles are then calculated from

$$\boldsymbol{F}\tilde{\boldsymbol{e}}_1 = 0 \quad \text{and} \quad \boldsymbol{F}^T\tilde{\boldsymbol{e}}_2 = 0, \tag{4.6}$$

where $\tilde{\boldsymbol{e}}_1 = [e_{11}, e_{12}, e_{13}]^T$ and $\tilde{\boldsymbol{e}}_1 = [e_{21}, e_{22}, e_{23}]^T$ are equal to the last column of $\boldsymbol{V}$ and $\boldsymbol{U}$ respectively. Then

$$x_i = e_{i1}/e_{i3} \quad \text{and} \quad y_i = e_{i2}/e_{i3} \quad \text{for} \quad i = 1, 2.$$

The four parameters (a, b, c, d) are found directly from the fundamental matrix $\boldsymbol{F}$. Thus the seven initial parameters are (x_1, y_1, x_2, y_2) and three among (a, b, c, d) and the final estimates are calculated by minimising the sum of distances between corner points and their epipolar lines. The following nonlinear equation is minimised:

$$\min_{\boldsymbol{F}} \sum_i d^2(\tilde{\boldsymbol{m}}_{2i}, \boldsymbol{F}\tilde{\boldsymbol{m}}_{1i}), \tag{4.7}$$

where

$$d(\tilde{\boldsymbol{m}}_2, \boldsymbol{F}\tilde{\boldsymbol{m}}_1) = \frac{|\tilde{\boldsymbol{m}}_2^T \boldsymbol{F}\tilde{\boldsymbol{m}}_1|}{\sqrt{(\boldsymbol{F}\tilde{\boldsymbol{m}}_1)_1^2 + (\boldsymbol{F}\tilde{\boldsymbol{m}}_1)_2^2}}$$

is the Euclidean distance of point $\boldsymbol{m}_2$ to its epipolar line $\boldsymbol{F}\tilde{\boldsymbol{m}}_1$, and $(\boldsymbol{F}\tilde{\boldsymbol{m}}_1)_i$ is the i^{th} variable of vector $\boldsymbol{F}\tilde{\boldsymbol{m}}_1$.

To make the calculation symmetric, equation (4.7) is extended to

$$\min_{\boldsymbol{F}} \sum_i \left(d^2(\tilde{\boldsymbol{m}}_{2i}, \boldsymbol{F}\tilde{\boldsymbol{m}}_{1i}) + d^2(\tilde{\boldsymbol{m}}_{1i}, \boldsymbol{F}^T\tilde{\boldsymbol{m}}_{2i})\right),$$

which can be rewritten by using the fact that $\tilde{\boldsymbol{m}}_2^T \boldsymbol{F}\tilde{\boldsymbol{m}}_1 = \tilde{\boldsymbol{m}}_1^T \boldsymbol{F}^T\tilde{\boldsymbol{m}}_2$:

$$\min_{\boldsymbol{F}} \sum_i \left(\frac{1}{(\boldsymbol{F}\tilde{\boldsymbol{m}}_{1i})_1^2 + (\boldsymbol{F}\tilde{\boldsymbol{m}}_{1i})_2^2} + \frac{1}{(\boldsymbol{F}^T\tilde{\boldsymbol{m}}_{2i})_1^2 + (\boldsymbol{F}^T\tilde{\boldsymbol{m}}_{2i})_2^2}\right)(\tilde{\boldsymbol{m}}_{2i}^T \boldsymbol{F}\tilde{\boldsymbol{m}}_{1i})^2. \tag{4.8}$$

4.4 Least-Median-of-Squares method

The above mentioned methods would introduce inaccuracies into the calculation of the fundamental matrix if outliers are present. The method outlined in this section is used in the corner matching process described in chapter 5 and it has the important ability to detect outliers or false matches and still give an accurate estimation of the fundamental matrix. This is originally based on the method outlined in chapter 5 of Rousseeuw and Leroy's book on regression [41] and adapted by Zhang et. al. for the fundamental matrix estimation [54, 52].

For n corner point correspondences $(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2i})$ as estimated in chapter 5, a *Monte Carlo* type technique is used to draw m subsamples of $p = 8$ different corner point correspondences. Then for each subsample j the fundamental matrix $\boldsymbol{F}_j$ is calculated. The median of squared residuals (M_j) is determined for each $\boldsymbol{F}_j$ with respect to the whole set of corner point correspondences:

$$M_j = \mathrm{med}_{i=1,\ldots,n}\left[d^2(\tilde{\boldsymbol{m}}_{2i}, \boldsymbol{F}_j\tilde{\boldsymbol{m}}_{1i}) + d^2(\tilde{\boldsymbol{m}}_{1i}, \boldsymbol{F}_j^T\tilde{\boldsymbol{m}}_{2i})\right].$$

The estimate of $\boldsymbol{F}_j$, for which M_j is minimal among all m M_j's, is kept for the next stage of the algorithm.

The number of subsamples m is determined by the following equation:

$$P = 1 - [1 - (1 - \varepsilon)^p]^m, \tag{4.9}$$

which calculates the probability that at least one of the m subsamples is 'good' (a 'good' subsample consists of p good correspondences) and assuming that the whole set of correspondences contains up to ε outliers. Zhang et. al. [54] sets the variables as follows: for $P = 0.99$ and assuming $\varepsilon = 40\%$, m will be equal to 272. To compensate for Gaussian noise, Rousseeuw and Leroy [41] calculate the *robust standard deviation* estimate

$$\tilde{\sigma} = 1.4826[1 + 5/(n - p)]\sqrt{M_j},$$

where M_j is the minimal median.

Based on the *robust standard deviation* $\tilde{\sigma}$, it is possible to assign a weight for each corner correspondence:

$$w_i = \begin{cases} 1 & \text{if } r_i^2 \leq (2.5\tilde{\sigma})^2 \\ 0 & \text{otherwise,} \end{cases}$$

where

$$r_i^2 = d^2(\tilde{\boldsymbol{m}}_{2i}, \boldsymbol{F}\tilde{\boldsymbol{m}}_{1i}) + d^2(\tilde{\boldsymbol{m}}_{1i}, \boldsymbol{F}^T\tilde{\boldsymbol{m}}_{2i}).$$

The outliers are therefore all the correspondences with weight $w_i = 0$ and are not taken into account. The fundamental matrix is then finally calculated by solving the weighted least-squares problem:

$$\min \sum_i w_i r_i^2. \tag{4.10}$$

All the nonlinear minimisations have been done using the *Levenberg-Marquardt* algorithm [35, 40], described in appendix C.

Bucket Technique

As mentioned before, a *Monte Carlo* type technique is implemented. A problem with this is that the eight points generated for each subsample could lie very close to each other which makes the estimation of the epipolar geometry very inaccurate. In order to achieve a more reliable result, the following *regularly random selection method*, developed by Zhang et. al. [54], is implemented.

This method is based on bucketing techniques. The minimum and maximum coordinates of the corner points in the first image define a region which is divided into $b \times b$ buckets, as seen in figure 4.1.

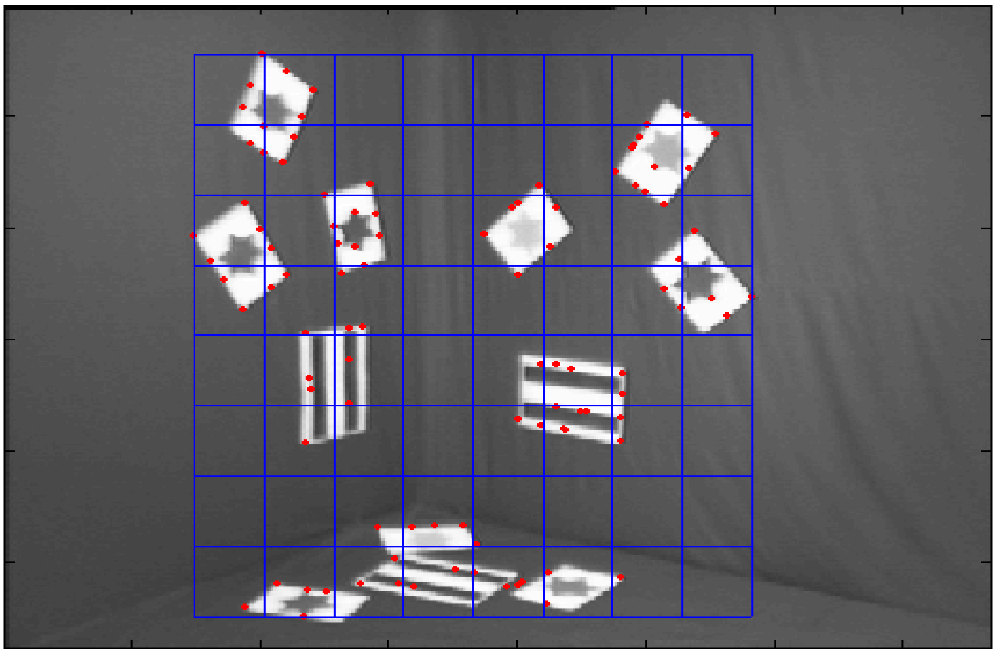


Figure 4.1: Illustration of bucketing technique.

Each bucket contains a number of corner points and indirectly a set of matches. Buckets having no matches are ignored. Then to generate a subsample, 8 mutually different buckets are randomly selected, and then one match is randomly selected in each bucket.

Due to a different number of matches in each bucket, a match in a bucket having only a few matches has a high probability of being selected. But it would be better if a bucket having many matches has the higher probability to be selected than a bucket having only few matches. This will guarantee that all matches have almost the same probability to be selected and this is achieved by the following: if a total of l buckets are available, then $[0,1]$ is divided into l intervals such that the width of the i^{th} interval is equal to $n_i / \sum_i n_i$, where n_i is the number of matches attached to the i^{th} bucket [54]. In the selection process, a number produced by a $[0,1]$ random generator and falling inside the i^{th} interval, implies that the i^{th} bucket has been selected. Figure 4.2 illustrates this process.

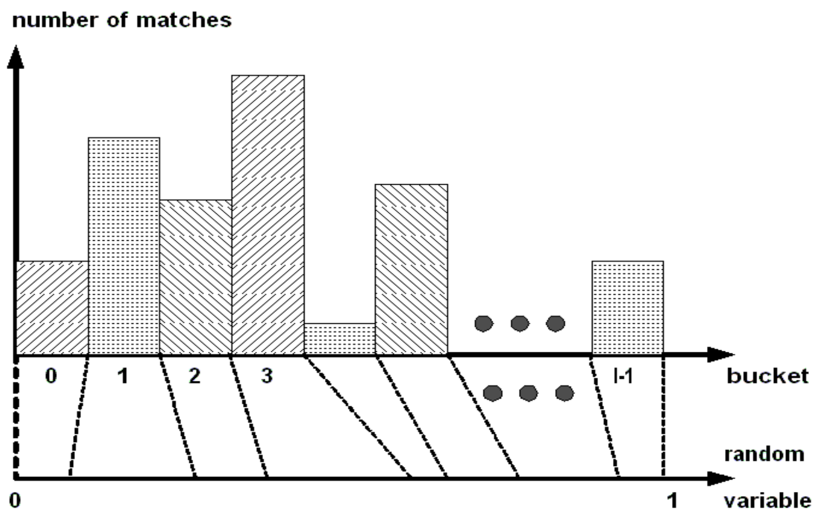


Figure 4.2: Interval and bucket mapping. Figure obtained from [54].

4.5 RANSAC

RANSAC or *random sample consensus* was first used for fundamental matrix estimation by Torr [47]. It is very similar to the LmedS method described above, a difference being that a threshold needs to be set by the user to determine if a feature pair is consistent with the fundamental matrix or not. This threshold is automatically calculated in the LMedS method. Instead of estimating the median of squared residuals, RANSAC calculates the size of the point matches that are consistent with each $\boldsymbol{F}_j$.

Zhang [52] mentions in his paper that if the fundamental matrix needs to be established for many images, then the LMedS method should be run on one pair of the images to find a suitable threshold, while RANSAC should be then run on all the remaining images, as RANSAC is able to terminate once a optimal solution is found and as such runs cheaper.

Chapter 5

Corner Matching

5.1 Introduction

Point matching plays an important part in the estimation of the fundamental matrix. Two different methods of point matching are introduced and combined to form a robust stereo matching technique. The first method [54] makes use of correlation techniques followed by relaxation methods. From these final correspondences, although not all perfect matches, the optimal fundamental matrix is calculated using the *Least-Median-of-Squares* method, which discards outliers or bad matches. The second method [36] sets up a proximity matrix weighted by the correlation between matches. Performing a singular value decomposition calculation on that matrix will ‘amplify’ good pairings and ‘attenuate’ bad ones.

Sections 5.2 and 5.3 of this chapter summarise the *correlation* and *strength of match measure* equations presented in the paper by Zhang et. al. [54], and calculate some correspondence between the corners in the two images. Section 5.4 describes the SVD algorithm by Pilu [36] and shows how to combine both methods to get a list of initial matches. In section 5.5 the stereo matching algorithm by Zhang et. al. [54] is outlined, which resolves false matches and outliers.

Results are given at the end of this chapter. The matching process works well on images containing different patterns and textures, and features are matched up perfectly under camera translation, rotation and zooming. However, if the image contains a repetitive pattern, features are not matched up at all.

It is impossible to obtain corner points from images that contain scenes with a uniform background. Therefore some markers need to be included in the scene in order to obtain sufficient corner points.

The corner extraction algorithm is described in appendix A.1.

5.2 Establishing Matches by Correlation

Corner points are represented by the vector $\boldsymbol{m}_i = [u_i, v_i]^T$ in the images. A correlation window of size $(2n+1) \times (2m+1)$ is centred at each corner detected in the first of two images. A rectangular search area of size $(2d_u+1) \times (2d_v+1)$ is placed around this point in the second image and for all the corners falling inside this area a correlation score is defined:

$$\text{Score}(\boldsymbol{m}_1, \boldsymbol{m}_2) = \frac{\sum_{i=-n}^{n} \sum_{j=-m}^{m} \left[I_1(u_1+i, v_1+j) - \overline{I_1(u_1, v_1)} \right] \times \left[I_2(u_2+i, v_2+j) - \overline{I_2(u_2, v_2)} \right]}{(2n+1)(2m+1)\sqrt{\sigma^2(I_1) \times \sigma^2(I_2)}}, \tag{5.1}$$

where $\overline{I_k(u,v)} = \sum_{i=-n}^{n} \sum_{j=-m}^{m} I_k(u+i, v+j)/[(2n+1)(2m+1)]$ is the average at point (u, v) of $I_k (k = 1, 2)$, and $\sigma(I_k)$ is the standard deviation of the image I_k in the neighbourhood $(2n+1) \times (2m+1)$ of (u, v), which is given by

$$\sigma(I_k) = \sqrt{\frac{\sum_{i=-n}^{n} \sum_{j=-m}^{m} I_k^2(u,v)}{(2n+1)(2m+1)} - \overline{I_k(u,v)}}. \tag{5.2}$$

The score ranges from -1 for uncorrelated windows to 1 for identical matches.

Matches above a certain threshold are then selected and form candidate matches. Thus each corner in the first image is associated with a set of candidate matches from the second image and vice versa. It is possible that there are no candidate matches for certain corners.

In this implementation, $n = m = 7$ for the correlation window and the threshold was chosen to be in the range of $0.7 - 0.8$. The search window size, d_u and d_v, was set to an eighth of the image width and height respectively.

5.3 Support of each Match

This section will define a measure of support for each match, which is called the strength of the match in [54].

Each candidate match is written as $(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j})$, where $\boldsymbol{m}_{1i}$ is a corner in the first image and $\boldsymbol{m}_{2j}$ a corner in the second image. Then $\mathcal{N}(\boldsymbol{m}_{1i})$ and $\mathcal{N}(\boldsymbol{m}_{2j})$ are the neighbours of $\boldsymbol{m}_{1i}$ and $\boldsymbol{m}_{2j}$ within a disc of radius R. If $(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j})$ is a good match, there should be many matches $(\boldsymbol{n}_{1k}, \boldsymbol{n}_{2l})$, where $\boldsymbol{n}_{1k} \in \mathcal{N}(\boldsymbol{m}_{1i})$ and $\boldsymbol{n}_{2l} \in \mathcal{N}(\boldsymbol{m}_{2j})$, such that the position of $\boldsymbol{n}_{1k}$ relative to $\boldsymbol{m}_{1i}$ is similar to that of $\boldsymbol{n}_{2l}$ relative to $\boldsymbol{m}_{2j}$. If the match is not so good, then there should be only a few matches in the neighbourhood or none at all.

The *strength of the match* or S is then defined as:

$$S(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j}) = c_{ij} \sum_{\boldsymbol{n}_{1k} \in \mathcal{N}(\boldsymbol{m}_{1i})} \left[\max_{\boldsymbol{n}_{2l} \in \mathcal{N}(\boldsymbol{m}_{2j})} \frac{c_{kl}\,\delta(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j}; \boldsymbol{n}_{1k}, \boldsymbol{n}_{2l})}{1 + \mathrm{dist}(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j}; \boldsymbol{n}_{1k}, \boldsymbol{n}_{2l})} \right], \tag{5.3}$$

where c_{ij} and c_{kl} are the correlation scores of the candidate matches $(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j})$ and $(\boldsymbol{n}_{1k}, \boldsymbol{n}_{2l})$ from the previous section. The average distance between the two pairings is defined as:

$$\mathrm{dist}(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j}; \boldsymbol{n}_{1k}, \boldsymbol{n}_{2l}) = [d(\boldsymbol{m}_{1i}, \boldsymbol{n}_{1k}) + d(\boldsymbol{m}_{2j}, \boldsymbol{n}_{2l})]/2$$

where $d(\boldsymbol{m}, \boldsymbol{n}) = \|\boldsymbol{m} - \boldsymbol{n}\|$ is the Euclidean distance between $\boldsymbol{m}$ and $\boldsymbol{n}$, and

$$\delta(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j}; \boldsymbol{n}_{1k}, \boldsymbol{n}_{2l}) = \begin{cases} e^{-r/\varepsilon_r} & \text{if } (\boldsymbol{n}_{1k}, \boldsymbol{n}_{2l}) \text{ is a candidate match and } r < \varepsilon_r \\ 0 & \text{otherwise,} \end{cases}$$

with r the relative distance difference given by

$$r = \frac{|d(\boldsymbol{m}_{1i}, \boldsymbol{n}_{1k}) - d(\boldsymbol{m}_{2j}, \boldsymbol{n}_{2l})|}{dist\,(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j}; \boldsymbol{n}_{1k}, \boldsymbol{n}_{2l})}$$

and ε_r a threshold on the relative distance difference.

The following points clarify the above equations:

- The strength of a match counts the number of candidate matches inside the neighbourhoods, but only those whose positions relative to the considered match are similar are counted.
- The similarity of relative positions is based on the relative distance r. When r is very big, the term $e^{-r/\varepsilon_r} \to 0$ and the candidate match $(\boldsymbol{n}_{1k}, \boldsymbol{n}_{2l})$ is ignored. When $r \to 0$, then $e^{-r/\varepsilon_r} \to 1$ and the candidate contributes largely to the match $(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j})$.
- If a corner point in the first image has several candidate matches in the second image, the one with the smallest distance difference is chosen as the final one using the ‘max’ operator.

- The contribution of a candidate match to the neighbourhood is weighted by its distance to the match. The '1' is added to prevent very close corner points from adding too much weight to the equation. This means that a close candidate match gives more support to the considered match than a distant one.

A problem pointed out by Zhang et. al. [54] is that the measure of match strength is not symmetrical. This means simply that the strength of a match is probably not the same if reversing the images, or $S(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j}) \neq S(\boldsymbol{m}_{2j}, \boldsymbol{m}_{1i})$. This happens when several corner points $\boldsymbol{n}_{1k} \in \mathcal{N}(\boldsymbol{m}_{1i})$ are candidate matches of a single corner point $\boldsymbol{n}_{2l} \in \mathcal{N}(\boldsymbol{m}_{2j})$, as shown in figure 5.1.

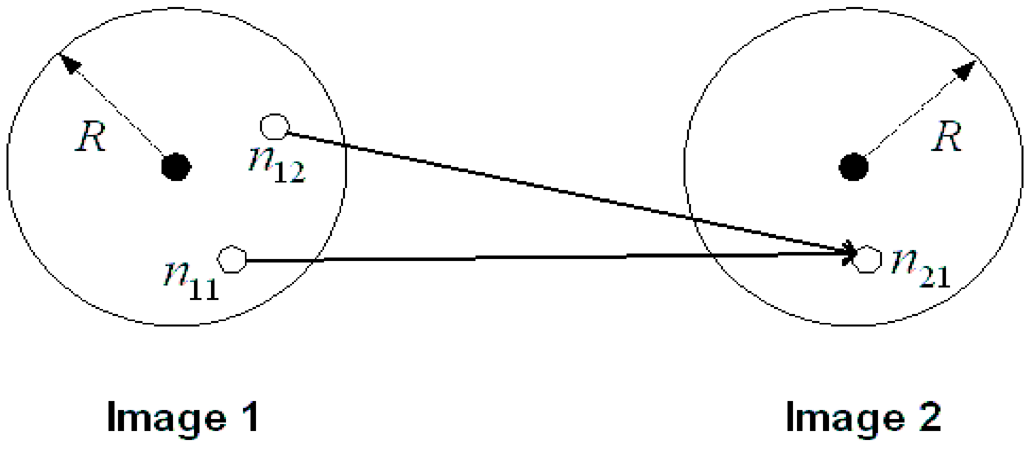


Figure 5.1: Non-symmetry problem for match strength. Figure obtained from [54].

To achieve symmetry, the following was suggested: before summing all the matches, if several corner points $\boldsymbol{n}_{1k} \in \mathcal{N}(\boldsymbol{m}_{1i})$ score the maximal value with the same corner point $\boldsymbol{n}_{2l} \in \mathcal{N}(\boldsymbol{m}_{2j})$, then only the corner point with the largest value is counted. This means that the same pairing will be counted if the images are reversed.

Another constraint which is added is that the angle of the rotation in the image plane must be below a certain value Θ. The idea here is that the angle between vector $\overrightarrow{\boldsymbol{m}_{1i}\boldsymbol{n}_{1k}}$ and vector $\overrightarrow{\boldsymbol{m}_{2j}\boldsymbol{n}_{2l}}$ must be less than Θ. For a candidate match where this constraint is not achieved, the value of $\delta(\boldsymbol{m}_{1i}, \boldsymbol{m}_{2j}; \boldsymbol{n}_{1k}, \boldsymbol{n}_{2l})$ is set to zero.

Zhang et. al. [54] set the following values as follows: R is equal to an eighth of the image width, $\varepsilon_r = 0.3$ and $\Theta = 90°$.

5.4 Initial Matches by Singular Value Decomposition

This section explains how initial matches are found via *Singular Value Decomposition,* a method described by Pilu [36] which is originally based on a paper by Scott and Longuet-Higgens [43]. At the end of this section it is shown how this technique can be combined with the strength measure of section 5.3.

This algorithm complies with Ullman's *minimal mapping theory* [48], which states three criteria for good global mapping:

1. the principle of similarity
2. the principle of proximity
3. the principle of exclusion.

The *principle of similarity* is an indication of how closely related the corner matches are. The *principle of proximity* states simply that if various corner matches are similar or equal, take the match which has the shortest distance between the two corner points. For the *principal of exclusion,* only a one-to-one mapping is allowed between corners.

The algorithm by Scott and Longuet-Higgins [43] will be extended to satisfy all these constraints.

The basic algorithm satisfies only the principle of proximity and exclusion. Having corner points $\boldsymbol{m}_{1i} (i = 1 \ldots m)$ in the first image and $\boldsymbol{m}_{2j} (j = 1 \ldots n)$ in the second image, a *proximity matrix* is set up as follows:

$$\boldsymbol{G}_{ij} = e^{-r_{ij}^2/2\sigma^2} \qquad i = 1 \ldots m, j = 1 \ldots n \tag{5.4}$$

where $r_{ij} = \|\boldsymbol{m}_{1i} - \boldsymbol{m}_{2j}\|$ is the Euclidean distance between the corner points if they are regarded of lying on the same plane. $\boldsymbol{G}_{ij}$ decreases from 1 to 0 with distance. The parameter σ controls the interaction between features: a small value of σ enforces local interactions while a large value allows for more global interactions.

The next stage of the algorithm is to perform *Singular Value Decomposition* (SVD) [15] of $\boldsymbol{G}$:

$$\boldsymbol{G} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^T,$$

where $\boldsymbol{U}$ and $\boldsymbol{V}$ are orthogonal matrices and $\boldsymbol{D}$ is the diagonal matrix containing the singular values along the diagonal elements $\boldsymbol{D}_{ii}$ in descending order.

The diagonal matrix $\boldsymbol{D}$ is then converted to a new matrix $\boldsymbol{E}$ which is obtained by replacing every diagonal element of $\boldsymbol{D}$ by a 1. Performing the following operation will then result in a new proximity matrix $\boldsymbol{P}$:

$$\boldsymbol{P} = \boldsymbol{U}\boldsymbol{E}\boldsymbol{V}^T.$$

The matrix $\boldsymbol{P}$ has the same shape as $\boldsymbol{G}$, but it has the property of pairing up good matches. Quoting Scott and Longuet-Higgens [43]:

> *"If $\boldsymbol{P}_{ij}$ is the greatest element in row i but not the greatest in column j, then we may regard $\mathbf{m}_{1i}$ as competing unsuccessfully for partnership with $\mathbf{m}_{2j}$; similar remarks apply if $\boldsymbol{P}_{ij}$ is the greatest element in its column but not in its row. But if $\boldsymbol{P}_{ij}$ is both the greatest element in its row and the greatest element in its column then we regard those features as being in 1:1 correspondence with one another".*

The *principle of proximity* arises from the nature of the *proximity matrix* and the *principle of exclusion* arises due to the orthogonality of matrix $\boldsymbol{P}$. The squares of the elements in each row of $\boldsymbol{P}$ can be added up to 1 and this implies that feature $\boldsymbol{m}_{1i}$ cannot be strongly associated with more than one feature $\boldsymbol{m}_{2j}$ [43].

To include the *principle of similarity* in the above algorithm, Pilu [36] adds in a measure of similarity, which is identical to the correlation score calculated in section 5.2. This is done in the following way:

$$\boldsymbol{G}_{ij} = \left[(c_{ij} + 1)/2\right] e^{-r_{ij}^2/2\sigma^2} \tag{5.5}$$

where c_{ij} is the correlation score defined in section 5.2. Matrix $\boldsymbol{G}$ is now called a *correlation-weighted proximity matrix* and still ranges from 0 to 1. The better the correlation between two features, the higher the value of $\boldsymbol{G}_{ij}$.

A great advantage of this algorithm is that it performs well when only a few features are available. Other algorithms, like the one described in Zhang et. al. [54], need many uniformly distributed features to be able to work properly. A disadvantage is the computational complexity and cost of the SVD for very large matrices. The implementation here makes use of only a maximum of 500 corners per image, which limits the proximity matrix to about 500 $\times$ 500 values.

At the end of his paper, Pilu [36] also suggests incorporating the strength of the match, as calculated in section 5.3, in the above algorithm. This could be achieved in various ways, either in place of the correlation score or in conjunction with equation (5.5).

Here we use the latter idea, and the following is an outline of how equation (5.5) is combined with the strength measure to select ‘good’ initial matches. Basically three criteria have to be met. If

1. $\boldsymbol{P}_{ij}$ is the greatest element in both row and column,
2. c_{ij} is greater than some correlation threshold (≈ 0.7),
3. and the considered match (i, j) has the greatest strength,

then this match will be included in the list of initial matches.

In effect, the above procedure replaces the relaxation process of Zhang et. al. [54], which consists of minimising an energy function summing up the strengths of all candidate matches.

5.5 Resolving False Matches

The method described in this section makes use of the epipolar geometry which exists between the two images in helping to resolve false matches. The fundamental matrix is calculated on the initial matched points estimated in section 5.4. The method used to calculate the fundamental matrix is the *Least-Median-of-Squares* (LMedS) method described in section 4.4, which discards false matches or outliers.

The initial matching algorithm described in sections 5.2 to 5.4 is then rerun on all corner points in both images, but the search window of section 5.2 is replaced by the epipolar constraint in the following way: the epipolar line for each corner point in the first image is calculated. If

$$\tilde{\boldsymbol{p}}_i = [\begin{array}{ccc} u_i & v_i & 1 \end{array}]$$

is a corner point in the first image in homogeneous coordinates, then its epipolar line is defined by

$$\boldsymbol{l}_i = \boldsymbol{F}\tilde{\boldsymbol{p}}_i,$$

where $\boldsymbol{F}$ is the fundamental matrix. Then the matched corner in the second image should, if it is a perfect match, lie on the epipolar line in the second image. A threshold determines whether possible points should be accepted or discarded. This threshold is in the form of a narrow band of width 2ϵ pixels centred on the epipolar line [54]. If a corner point falls within this band, it is accepted.

The value of ϵ is chosen to be $3.8\overline{d}$ for a probability of 95%, where $\overline{d}$ is the root of mean squares of distances between the corner points and their epipolar lines defined by the recovered fundamental matrix [54]:

$$\overline{d} = \sqrt{\sum_i w_i r_i^2 / \sum_i w_i}.$$

For a better understanding of the variables in the above equation, refer to section 4.4.

The fundamental matrix can now be refined after the second matching process, and matches which are still not close enough to the epipolar line are discarded.

5.6 Results

The matching algorithm can be summarised as follows: corner points established in each image independently are matched up using a correlation technique to find initial matches. Making use of the robust Least-Median-of-Squares method described in section 4.4, which takes the epipolar geometry between the two images into account, it is possible to determine false matches in the initial group of matched corners. The algorithm is then rerun, but this time the epipolar geometry is used to select the final matched corners.

Good results have been achieved by the above algorithm. It only fails if the images contain a repetitive pattern, as seen in figure 5.2, where all matches are incorrect.

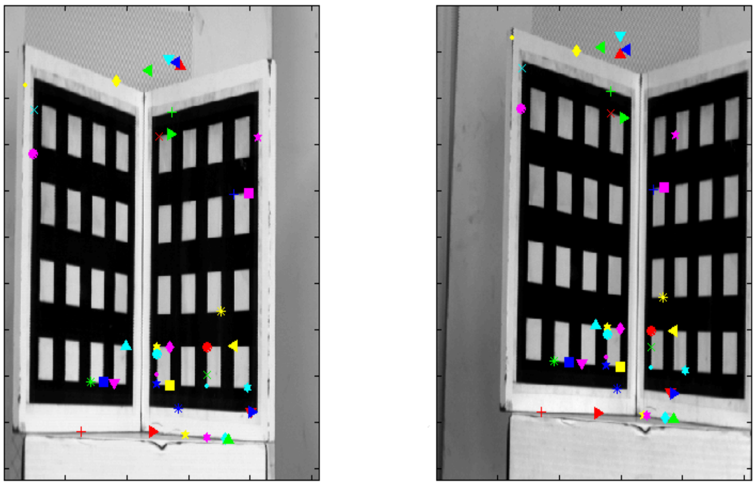
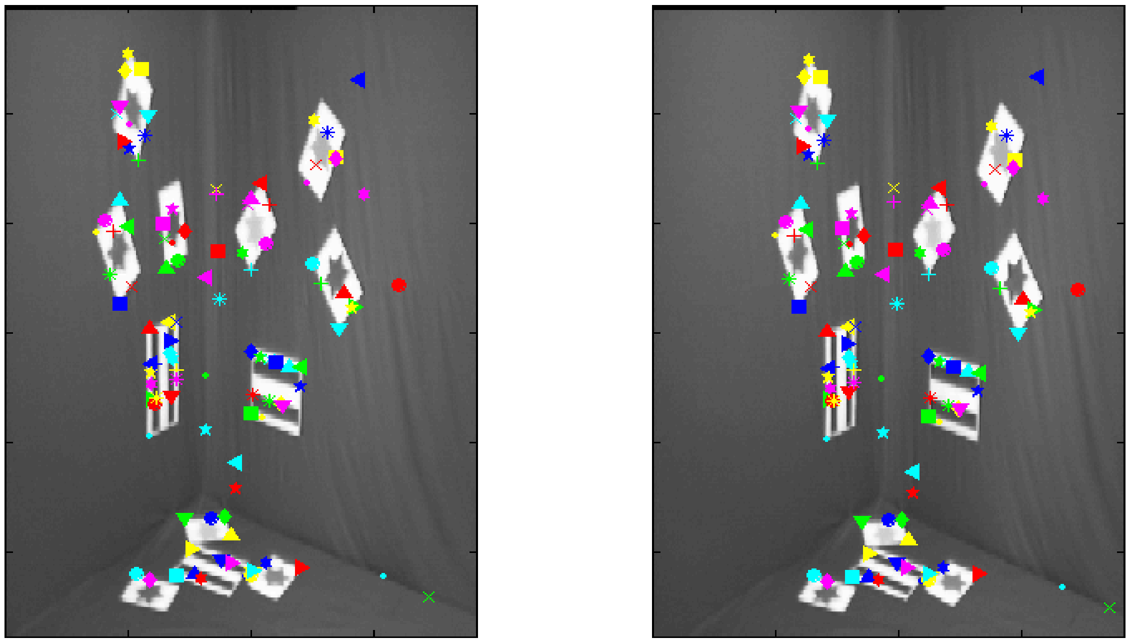


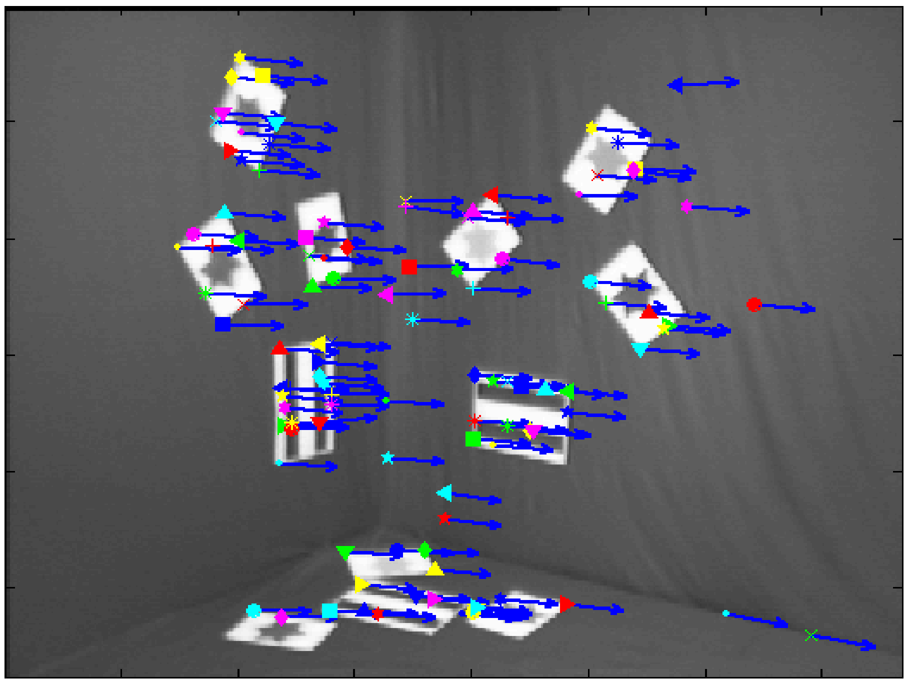
Figure 5.2: Repetitive Pattern (*INRIA-Robotvis project*).

Due to the fact that all corners of the checkered pattern have nearly the same correlation, it is nearly impossible to match up the corners perfectly. The corner points in each image are represented by a specific pattern and colour. In this way matched points are easily identified. Better results can be achieved with a repetitive pattern if it is placed in a scene containing different structures and patterns.

For a scene with a uniform background, some markers need to be put up in order to find corresponding corners. This is seen in figure 5.3(a). Here all corners have been perfectly matched. Figure 5.3(b) shows the camera translation (arrows point to the right) between the two images. The matching algorithm also has a 100% success rate for the same scene under camera rotation (arrows point in clockwise direction) and zooming (arrows point inwards), as seen in figure 5.4 and 5.5.

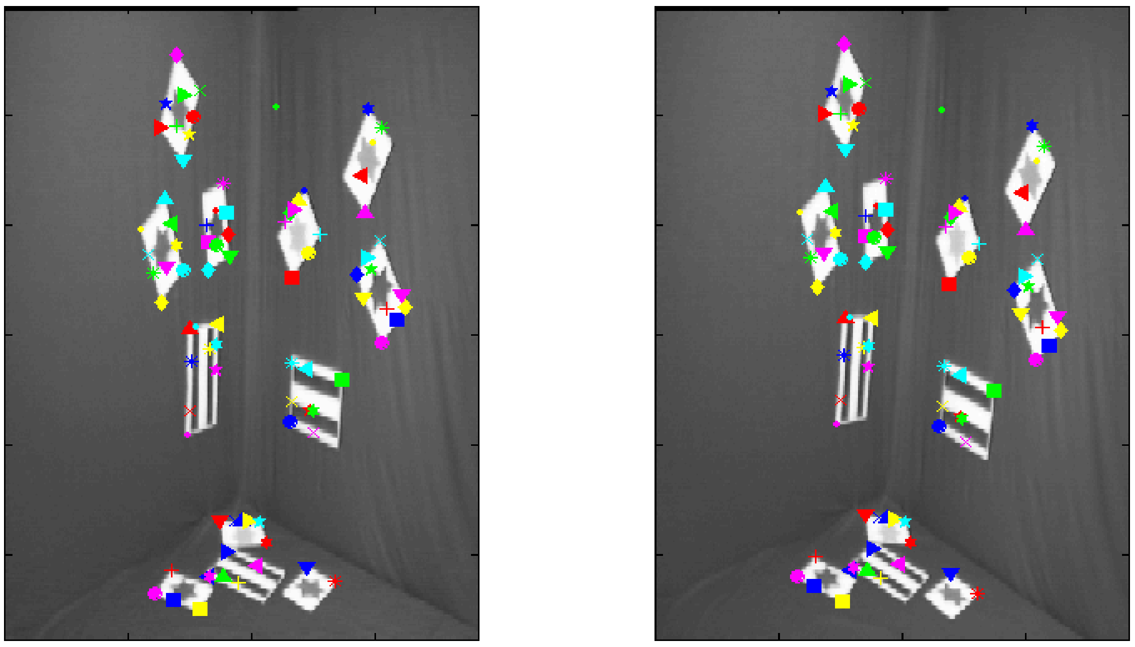


(a)

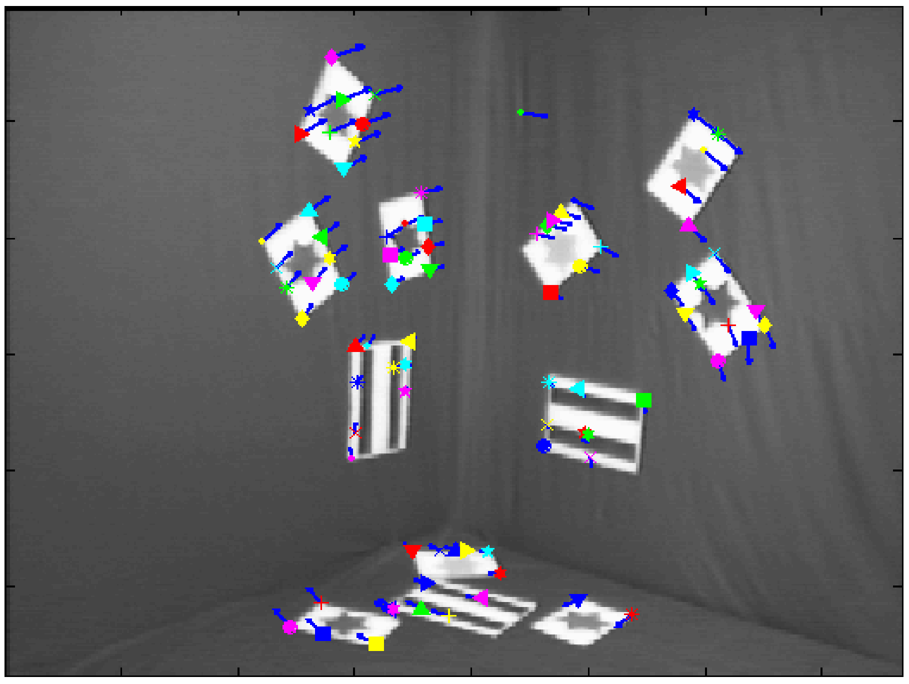


(b)

Figure 5.3: Uniform background scene with markers (Camera Translation).

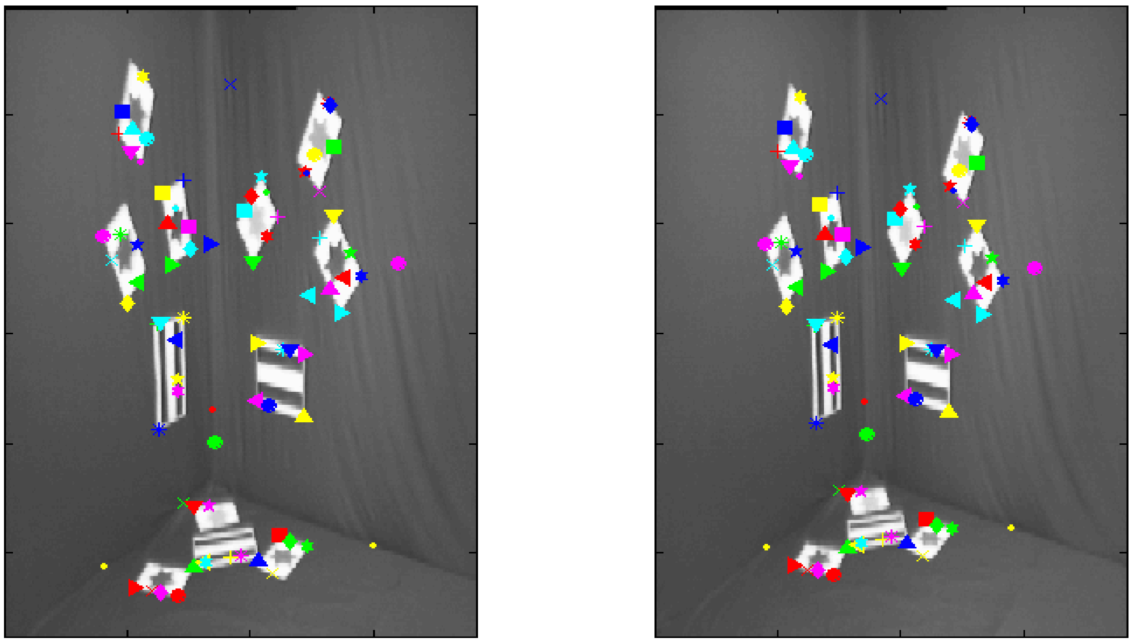


(a)

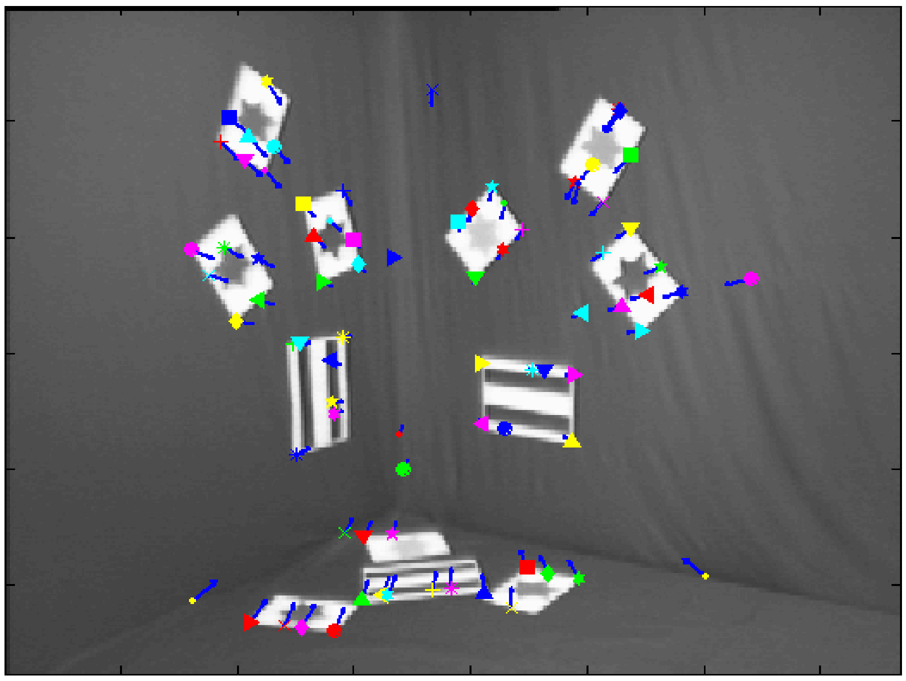


(b)

Figure 5.4: Uniform background scene with markers (Camera Rotation).



(a)



(b)

Figure 5.5: Uniform background scene with markers (Camera Zooming).

Chapter 6

Camera Calibration

6.1 Introduction

This chapter covers various camera calibration techniques.

Calibration is a fundamental property of 3D reconstruction. Usually the internal parameters of each camera are very accurately known beforehand and the whole environment is highly controlled, or a calibration object in the scene is used to calibrate the camera. But in many situations the source of the images is not known, which means that the camera's internal parameters are also not known, or it is desirable to change a camera midway through an image application. This means that the internal parameters of the camera can only be extracted from the images themselves.

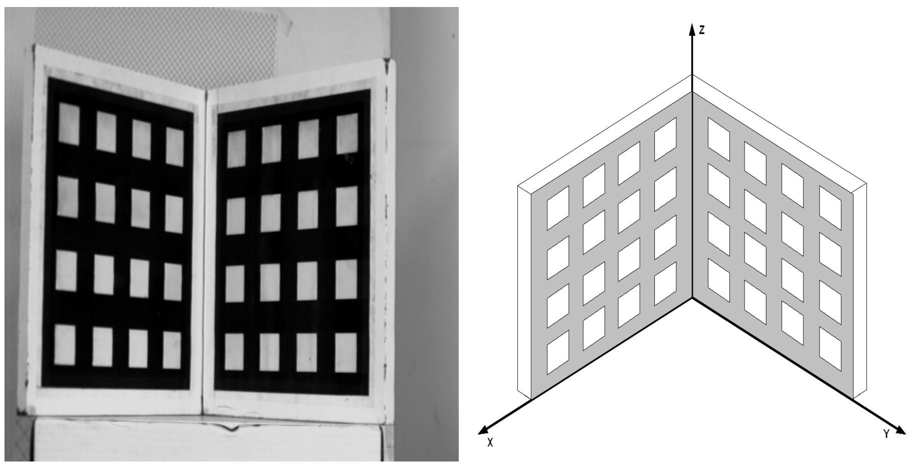
Section 6.2 gives a background to calibration, explaining original calibration methods. Section 6.3 describes camera selfcalibration and the theory behind *Kruppa's equations*. In section 6.4 selfcalibration is explained in terms of scene and auto-calibration constraints from only a single image.

A planar object is used in section 6.5 to estimate the camera calibration matrix. The planar pattern consists of a grid pattern, which is imaged from different points of view. From each view, corner points are extracted in order to calculate the correspondence between the image plane and the planar object. The correspondence is in the form of a homography matrix. Then for each view, a homography is established and allows for camera calibration.

6.2 Classical Calibration Methods

6.2.1 Introduction

The classical calibration method makes use of a calibration pattern of known size inside the view of the camera. Sometimes this will be a flat plate with a regular pattern marked on it [32] (see figure 6.1(a)) or a scene containing some control points with known coordinates [29]. A disadvantage of these methods is that it is impossible to calibrate a camera while it is involved in some imaging task. If any change in the camera's settings occur, a correction is not possible without interrupting the task. The change of the camera's settings may be a change in the focal length, or small mechanical or thermal changes affecting the camera as a whole.



(a) Calibration Pattern. (b) Coordinate System of Pattern.

Figure 6.1: Common Calibration Pattern.

As seen in figure 6.1, two flat planes are assembled with an angle of 90° between them. These two planes define a coordinate system as seen in figure 6.1(b). The coordinates of the corners of the white squares on the planes are known in terms of this coordinate system. It is then relatively easy to extract those corners in the image, and the correspondence between the 3D points and the 2D image points gives a projective map from $\mathcal{P}^3 \rightarrow \mathcal{P}^2$, which is the perspective projection matrix $\boldsymbol{P}$ mentioned in equation (3.1). Having calculated this projection matrix, it can be decomposed into the form of equation (3.8) by means of *QR decomposition* [1].

6.2.2 Estimating the Perspective Projection Matrix

By minimising the image error, the perspective projection matrix is estimated for n 3D points $\boldsymbol{M}_i$ corresponding to image points $\boldsymbol{m}_i$. This image error is the distance between the actual image point and the projection of the world point onto the image plane using $\boldsymbol{P}$ [1].

Making use of equation (3.1), with $\tilde{\boldsymbol{m}} = [u, v, 1]^T$ and $\tilde{\boldsymbol{M}} = [X, Y, Z, 1]^T$, three equations can be obtained, but dividing by the third one gives two equations in the 12 unknown parameters of $\boldsymbol{P}$:

$$\begin{aligned} u &= \frac{P_{11}X + P_{12}Y + P_{13}Z + P_{14}}{P_{31}X + P_{32}Y + P_{33}Z + P_{34}} \\ v &= \frac{P_{21}X + P_{22}Y + P_{23}Z + P_{24}}{P_{31}X + P_{32}Y + P_{33}Z + P_{34}}. \end{aligned} \tag{6.1}$$

The function which needs to be minimised is defined as the squared geometric distance between the actual image points and the projected image points:

$$\begin{aligned} E_g = \frac{1}{n}\sum_{i=1}^{n} \Bigg[&\left(u_i - \frac{P_{11}X + P_{12}Y + P_{13}Z + P_{14}}{P_{31}X + P_{32}Y + P_{33}Z + P_{34}} \right)^2 + \\ &\left(v_i - \frac{P_{21}X + P_{22}Y + P_{23}Z + P_{24}}{P_{31}X + P_{32}Y + P_{33}Z + P_{34}} \right)^2 \Bigg]. \end{aligned} \tag{6.2}$$

The above error function is non-linear and can be minimised using the Levenberg-Marquardt minimisation algorithm described in appendix C. Between iterations, the matrix $\boldsymbol{P}$ is usually scaled ($\|\boldsymbol{P}\| = 1$) or one parameter of $\boldsymbol{P}$ can be fixed ($P_{34} = 1$).

To find an initial estimate, the equations of (6.1) are rearranged, so that instead of minimising the geometric distance E_g, an algebraic distance E_a is minimised [1]:

$$\begin{aligned} E_a = \frac{1}{n}\sum_{i=1}^{n} \big[&(u_i(P_{31}X + P_{32}Y + P_{33}Z + P_{34}) - (P_{11}X + P_{12}Y + P_{13}Z + P_{14}))^2 + \\ &(v_i(P_{31}X + P_{32}Y + P_{33}Z + P_{34}) - (P_{21}X + P_{22}Y + P_{23}Z + P_{24}))^2 \big]. \end{aligned} \tag{6.3}$$

This error function is linear in the unknown parameters of $\boldsymbol{P}$ and can be rearranged into the following form:

$$\min_{p} \|\boldsymbol{Z}\boldsymbol{p}\|^2, \tag{6.4}$$

subject to $\|\boldsymbol{p}\|^2 = 1$. The vector $\boldsymbol{p}$ is a column vector of the elements of the perspective projection matrix $\boldsymbol{P}$, and the matrix $\boldsymbol{Z}$ is defined as:

$$\boldsymbol{Z} = \begin{bmatrix} \tilde{\boldsymbol{M}}_1^T & \boldsymbol{0}^T & -u_1\tilde{\boldsymbol{M}}_1^T \\ \boldsymbol{0}^T & \tilde{\boldsymbol{M}}_1^T & -v_1\tilde{\boldsymbol{M}}_1^T \\ \tilde{\boldsymbol{M}}_2^T & \boldsymbol{0}^T & -u_2\tilde{\boldsymbol{M}}_2^T \\ \vdots & \vdots & \vdots \\ \boldsymbol{0}^T & \tilde{\boldsymbol{M}}_n^T & -v_n\tilde{\boldsymbol{M}}_n^T \end{bmatrix}.$$

The solution of equation (6.4) is then the eigenvector of $\boldsymbol{Z}$ corresponding to the smallest eigenvalue, and can be found via *Singular Value Decomposition* [15].

6.2.3 Extracting the Camera Calibration Matrix

Once the perspective projection matrix $\boldsymbol{P}$ has been estimated, it can be decomposed into the form of equation (3.8). The following 3×3 submatrix of $\boldsymbol{P}$ can be expressed as follows:

$$\begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \end{bmatrix} = \boldsymbol{K}\boldsymbol{R},$$

where matrix $\boldsymbol{K}$ is the camera calibration matrix which is upper triangular and $\boldsymbol{R}$ is orthogonal. Armstrong [1] uses *QR decomposition* [15] to find $\boldsymbol{K}$ and $\boldsymbol{R}$.

6.2.4 Other Methods

Photogrammetrist make use of other methods to calibrate their cameras. A calibration grid with markers is also used as before, as seen in figure 6.2. The 3D coordinates of the markers are known, and methods such as *Direct Linear Transformation* and *Bundle Adjustment* are used to accurately estimate the internal parameters of the camera. Usually more than one image of the same calibration object with different orientations is used. Commercial software such as the *Australis* program from the University of Melbourne performs these calculations and produces highly accurate results. This program is used to find the real calibration matrix of the camera for a particular setting and the results are compared to the calibration method outlined in section 6.5.

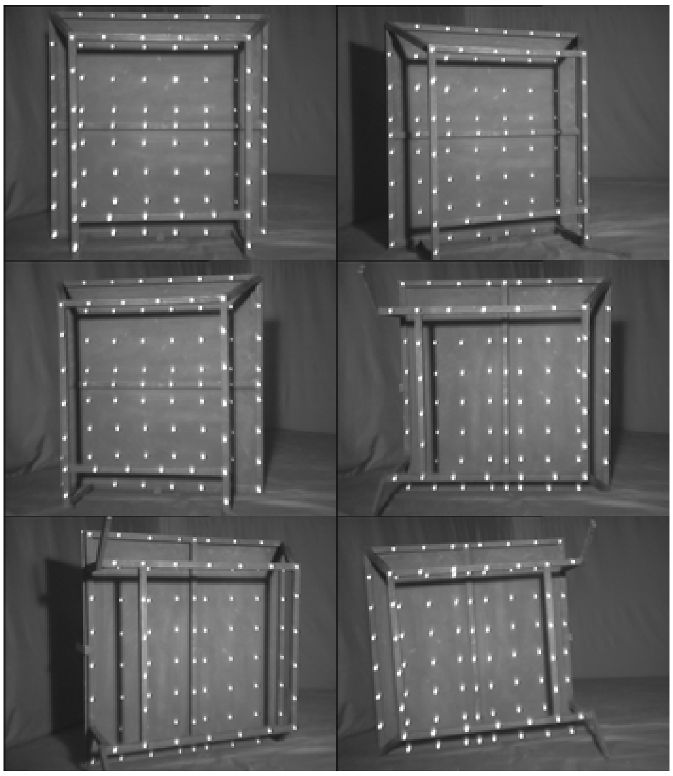


Figure 6.2: Six images of a calibration object from the department of geomatics at UCT.

6.3 Selfcalibration using Kruppa's Equations

6.3.1 Introduction

In the case of selfcalibration, the known object in the scene is replaced by an abstract object, the *absolute conic* mentioned in chapter 2. The absolute conic is a particular conic in the plane of infinity, which is invariant to transformations of 3D space (see figure 6.3). This means that the image of the absolute conic $\boldsymbol{\omega}_\infty$ is independent of the position and orientation of the camera. In figure 6.3, if the camera moves from position C_1 to position C_2 and provided the internal parameters of the camera stay constant, the image of the absolute conic will be the same in both image planes.

The image of the absolute conic is related to the camera calibration matrix in the following way:

$$\boldsymbol{\omega}_\infty = \boldsymbol{K}^{-T}\boldsymbol{K}^{-1}. \tag{6.5}$$

The calibration matrix can then be extracted from $\boldsymbol{\omega}_\infty$ by *Cholesky decomposition* [15]. Thus knowing $\boldsymbol{\omega}_\infty$ is equivalent of knowing $\boldsymbol{K}$. Note that $\boldsymbol{\omega}_\infty$ is a symmetric matrix. The following section will go into more detail on how to solve for $\boldsymbol{\omega}_\infty$.

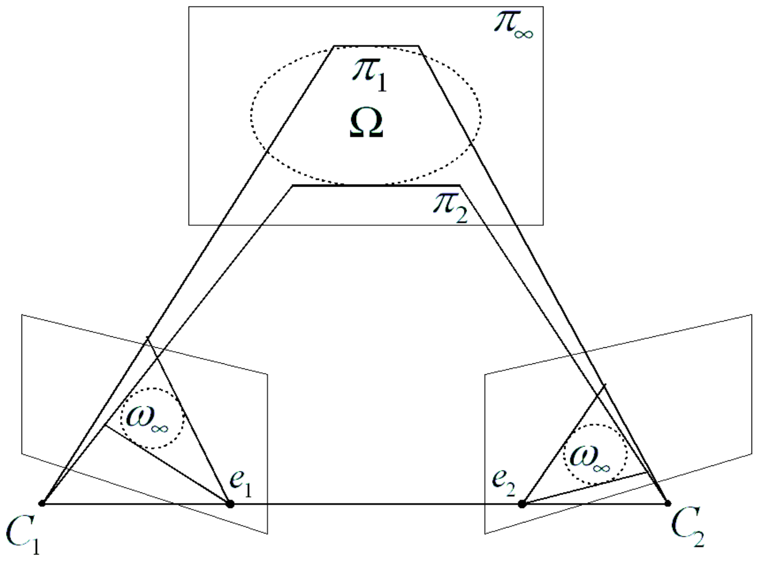


Figure 6.3: The Image of the Absolute Conic.

6.3.2 Kruppa's Equations

Kruppa's equations link the epipolar transformation or the fundamental matrix to the image of the absolute conic $\boldsymbol{\omega}_\infty$. Three epipolar transformations arising from three different camera motions are enough to determine $\boldsymbol{\omega}_\infty$ [10]. A description of Kruppa's equations is given in the next section based on a paper by Lourakis and Deriche [30], and some new developments in the estimation of the parameters in Kruppa's equations are outlined.

Description of Kruppa's Equations

Each point $\boldsymbol{p}$ belonging to the image of the absolute conic $\boldsymbol{\omega}_\infty$ in the second image satisfies $\tilde{\boldsymbol{p}}^T \boldsymbol{\omega}_\infty \tilde{\boldsymbol{p}} = 0$. Figure 6.3 also shows two planes, $\boldsymbol{\pi}_1$ and $\boldsymbol{\pi}_2$, which are tangent to the absolute conic $\boldsymbol{\Omega}$ and pass through the two camera centres. This plane intersects the image planes at two pairs of epipolar lines, which are tangent to the image of the absolute conic $\boldsymbol{\omega}_\infty$. Tangent lines to conics are better expressed in terms of dual conics [33, 30]. The dual conic defines the locus of lines to the original conic and is given by the inverse of the original conic matrix.

Thus the dual of the image of the absolute conic $\boldsymbol{\omega}_\infty^*$ is defined as:

$$\boldsymbol{\omega}_\infty^* = \boldsymbol{K}\boldsymbol{K}^T, \tag{6.6}$$

and therefore $\boldsymbol{l}^T\boldsymbol{\omega}_\infty^*\boldsymbol{l} = 0$, where line $\boldsymbol{l}$ is tangent to $\boldsymbol{\omega}_\infty$. Then it can be shown that a point $\boldsymbol{q}$ on any of the tangents to $\boldsymbol{\omega}_\infty$ in the second image will satisfy

$$(\boldsymbol{e}_2 \times \tilde{\boldsymbol{q}})^T\boldsymbol{\omega}_\infty^*(\boldsymbol{e}_2 \times \tilde{\boldsymbol{q}}) = 0$$

The term $\boldsymbol{F}^T\tilde{\boldsymbol{q}}$ is the epipolar line corresponding to $\boldsymbol{q}$ in the first image and is also tangent to $\boldsymbol{\omega}_\infty$. Because of the invariance of $\boldsymbol{\omega}_\infty$ under any transformations, the following equation is obtained:

$$(\boldsymbol{F}^T\tilde{\boldsymbol{q}})^T\boldsymbol{\omega}_\infty^*(\boldsymbol{F}^T\tilde{\boldsymbol{q}}) = 0.$$

Combining the above two equations will yield:

$$\boldsymbol{F}\boldsymbol{\omega}_\infty^*\boldsymbol{F}^T = \beta([\boldsymbol{e}_2]_x)^T\boldsymbol{\omega}_\infty^*[\boldsymbol{e}_2]_x = \beta[\boldsymbol{e}_2]_x\boldsymbol{\omega}_\infty^*([\boldsymbol{e}_2]_x)^T, \tag{6.7}$$

where β is an arbitrary, nonzero scale factor and $[\boldsymbol{e}_2]_x$ is the antisymmetric matrix of vector $\boldsymbol{e}_2$ as described in equation (2.6).

To explain equation (6.7) in words: "*the Kruppa equations express the constraint that epipolar lines in the second image that correspond to epipolar lines of the first image that are tangent to $\boldsymbol{\omega}_\infty$, are also tangent to $\boldsymbol{\omega}_\infty$ and vice versa*" [30].

As $\boldsymbol{F}\boldsymbol{\omega}_\infty^*\boldsymbol{F}^T$ is a symmetric matrix, equation (6.7) corresponds to the following equations obtained by eliminating β:

$$\begin{aligned} &\frac{\boldsymbol{F}\boldsymbol{\omega}_\infty^*\boldsymbol{F}_{11}^T}{([\boldsymbol{e}_2]_x\boldsymbol{\omega}_\infty^*([\boldsymbol{e}_2]_x)^T)_{11}} = \frac{\boldsymbol{F}\boldsymbol{\omega}_\infty^*\boldsymbol{F}_{12}^T}{([\boldsymbol{e}_2]_x\boldsymbol{\omega}_\infty^*([\boldsymbol{e}_2]_x)^T)_{12}} = \frac{\boldsymbol{F}\boldsymbol{\omega}_\infty^*\boldsymbol{F}_{13}^T}{([\boldsymbol{e}_2]_x\boldsymbol{\omega}_\infty^*([\boldsymbol{e}_2]_x)^T)_{13}} = \\ =&\frac{\boldsymbol{F}\boldsymbol{\omega}_\infty^*\boldsymbol{F}_{22}^T}{([\boldsymbol{e}_2]_x\boldsymbol{\omega}_\infty^*([\boldsymbol{e}_2]_x)^T)_{22}} = \frac{\boldsymbol{F}\boldsymbol{\omega}_\infty^*\boldsymbol{F}_{23}^T}{([\boldsymbol{e}_2]_x\boldsymbol{\omega}_\infty^*([\boldsymbol{e}_2]_x)^T)_{23}} = \frac{\boldsymbol{F}\boldsymbol{\omega}_\infty^*\boldsymbol{F}_{33}^T}{([\boldsymbol{e}_2]_x\boldsymbol{\omega}_\infty^*([\boldsymbol{e}_2]_x)^T)_{33}} \end{aligned} \tag{6.8}$$

There are only two independent equations among the six equations of (6.8) [30]. These equations are second order polynomials in the elements of $\boldsymbol{\omega}_\infty^*$.

The Simplified Kruppa's Equations

In recent years a simplified approach to solving Kruppa's equations has been developed, which makes use of the *Singular Value Decomposition* of the fundamental matrix $\boldsymbol{F}$ and is described in detail by Lourakis and Deriche [30] and based on the paper by Hartley [19]. With this method the equations of (6.8) are reduced to three and are independent of the epipole $\boldsymbol{e}_2$. But the actual solving process is still very complex, as individual steps in the calculation involve expanding matrices through differentiation and having to estimate the variance of the vector containing the parameters of the *Singular Value Decomposition* of the fundamental matrix. Lourakis and Deriche [30] mention that they are still working on a technique which would simplify this whole process, especially the calculation of the variance mentioned above.

6.4 Selfcalibration in Single Views

6.4.1 Introduction

This section describes a way of combining image, scene and auto-calibration constraints for calibration of single views. The method is not limited to single views, but for the purpose of explaining the theory, a single view is sufficient. The advantage of this method over Kruppa's equations is that the equations obtained here are linear and therefore the solution is very easily calculated. The disadvantage is that some sort of calibration pattern is needed and calibration needs to be done independently of the 3D reconstruction.

The method is especially suited for building architectural models [29], as buildings contain mostly planes and lines in orthogonal directions which are important to this calibration method. Any room contains three planes orthogonal to each other, i.e. two walls and the floor, and thus this method is perfect for calibrating cameras in a room.

Two methods of calibration of a single view are outlined, both based on descriptions by Liebowitz et. al. [27, 28, 29].

6.4.2 Some Background

As mentioned in section 2.4.1, there are two points which lie on the line at infinity. These points are called circular points, which are a complex conjugate point pair ($\boldsymbol{x} = [1, \pm i, 0]^T$ in a metric coordinate frame) and are also invariant to similarity transformations (rotations and translations) of space. They arise from the following: a plane in space intersects the plane at infinity $\boldsymbol{\pi}_\infty$ in the line at infinity $\boldsymbol{l}_\infty$ which intersects the absolute conic in two points, which are the circular points. A vanishing line of a space plane intersects the image of the absolute conic $\boldsymbol{\omega}_\infty$ in two points, which are the imaged circular points.

Knowing that a world plane is mapped to the image plane by a projective transformation or homography ($\boldsymbol{H}$) [28], the imaged circular points are defined as

$$\boldsymbol{I} = \boldsymbol{H}^{-1}[1, i, 0]^T = [\alpha - i\beta, 1, -l_2 - \alpha l_1 + i l_1 \beta]^T \tag{6.9}$$

and $\boldsymbol{J} = \mathrm{conj}(\boldsymbol{I})$. Additionally,

$$\boldsymbol{H} = \boldsymbol{SAP},$$

with

$$\boldsymbol{S} = \begin{bmatrix} s\boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}_3^T & 1 \end{bmatrix},$$

where $\boldsymbol{R}$ is the rotation matrix, $\boldsymbol{t}$ the translation vector and s a scale factor. The matrix

$$\boldsymbol{P} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ l_1 & l_2 & l_3 \end{bmatrix}, \tag{6.10}$$

where $\boldsymbol{l}_\infty = [l_1, l_2, l_3]^T$ is the vanishing line of the plane. Usually $\boldsymbol{l}_\infty$ is normalised, such that $l_3 = 1$. Finally,

$$\boldsymbol{A} = \begin{bmatrix} \frac{1}{\beta} & -\frac{\alpha}{\beta} & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{6.11}$$

where β and α define the image of the circular points [27, 28]. Matrix $\boldsymbol{S}$ represents a similarity transformation, $\boldsymbol{A}$ an affine transformation and $\boldsymbol{P}$ a projective transformation. The homography matrix $\boldsymbol{H}$ is stratified into the 3 geometries. This is similar to the description of the stratification of 3D space in chapter 2, but in this section only the 2D space is stratified.

6.4.3 Calibration Method 1

The first method presented here requires three planes in the image to be orthogonal to each other. In a room, this could be two walls and the ground plane, as seen in figure 6.4. To be able to extract many features from these planes, a calibration pattern is placed on each plane, such that they are also orthogonal to each other.

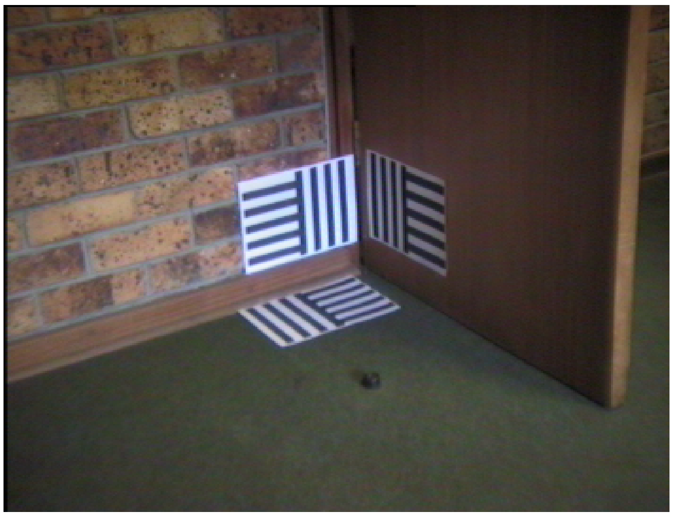
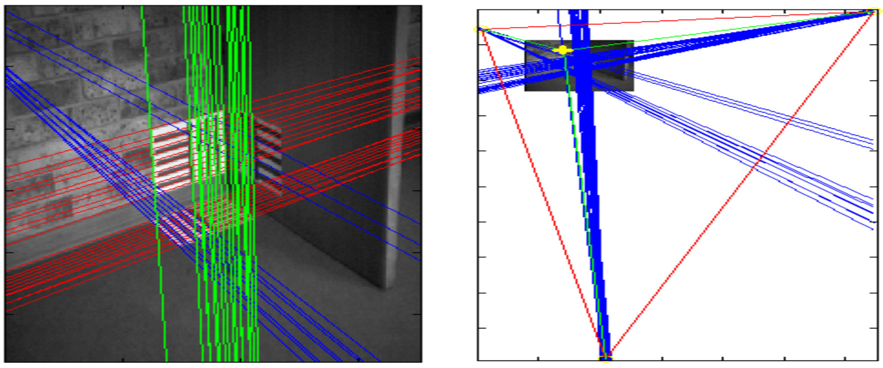


Figure 6.4: Image illustrating three orthogonal planes.

It is then necessary to extract all parallel lines in three orthogonal directions from the image. This is done with the help of the line algorithm of Burns et. al., as described in Appendix A.2. The lines extracted using this algorithm are seen in figure 6.5(a).



(a) Parallel Lines in three orthogonal directions.

(b) Triangle with vanishing points as vertices and showing the principal point in yellow (orthocentre of image).

Figure 6.5: Three Orthogonal Vanishing Points.

For all the parallel lines in each orthogonal direction, the vanishing point is calculated, i.e. the intersection of all parallel lines in the same direction is found. Appendix B outlines a accurate method of calculating vanishing points from a set of parallel lines.

The following assumption is then made: if the camera has a unit aspect ratio and zero skew, the principal point of the camera will be at the orthocentre of the triangle formed by the three vanishing points [27, 29, 5]. This can be seen in figure 6.5(b).

The principal point is calculated by combining a set of constraints: writing the image of the absolute conic as follows:

$$\boldsymbol{\omega}_\infty = \begin{bmatrix} \omega_1 & \omega_2 & \omega_4 \\ \omega_2 & \omega_3 & \omega_5 \\ \omega_4 & \omega_5 & \omega_6 \end{bmatrix} \tag{6.12}$$

and the three vanishing points as $\boldsymbol{u} = [u_1, u_2, u_3]^T$, $\boldsymbol{v} = [v_1, v_2, v_3]^T$ and $\boldsymbol{w} = [w_1, w_2, w_3]^T$, then the following three constraints arise:

$$\begin{aligned} \boldsymbol{u}^T \boldsymbol{\omega}_\infty \boldsymbol{v} &= 0 \\ \boldsymbol{u}^T \boldsymbol{\omega}_\infty \boldsymbol{w} &= 0 \\ \boldsymbol{v}^T \boldsymbol{\omega}_\infty \boldsymbol{w} &= 0. \end{aligned} \tag{6.13}$$

This means that a pair of vanishing points arising from orthogonal directions are conjugate with respect to $\boldsymbol{\omega}_\infty$ [27]. (See also section 2.2.2 on poles and polars.)

Expanding the first equation to

$$u_1v_1\omega_1 + (u_1v_2 + u_2v_1)\omega_2 + u_2v_2\omega_3 + (u_1v_3 + u_3v_1)\omega_4 + (u_2v_3 + u_3v_2)\omega_5 + u_3v_3\omega_6 = 0 \tag{6.14}$$

and then writing the elements of $\boldsymbol{\omega}_\infty$ as a vector

$$\boldsymbol{\omega}_v = (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6)^T$$

and the coefficients of the elements of $\boldsymbol{\omega}_v$ as

$$\boldsymbol{\kappa}_{uv}^T = (u_1v_1, u_1v_2 + u_2v_1, u_2v_2, u_1v_3 + u_3v_1, u_2v_3 + u_3v_2, u_3v_3)^T,$$

the linear constraint is written as:

$$\boldsymbol{\kappa}_{uv}^T \boldsymbol{\omega}_v = 0. \tag{6.15}$$

Thus for each pair of orthogonal vanishing points, an additional constraint is obtained. Assuming a unit aspect ratio and zero skew for the camera, $\omega_2 = 0$ and $\omega_1 - \omega_3 = 0$ are two additional constraints. These two constraints can be found by expanding $\boldsymbol{\omega}_\infty$ in terms of the parameters of the calibration matrix $\boldsymbol{K}$. This means for three vanishing points and the two assumptions from above, five constraints are obtained. This will determine $\boldsymbol{\omega}_\infty$ and then $\boldsymbol{K}$.

A coefficient matrix $\boldsymbol{A}$ is set up from equation (6.15) and the two constraints obtained from the above mentioned assumptions:

$$\boldsymbol{A}^T = \begin{bmatrix} u_1v_1 & u_1w_1 & v_1w_1 & 0 & 1 \\ u_1v_2 + u_2v_1 & u_1w_2 + u_2w_1 & v_1w_2 + v_2w_1 & 1 & 0 \\ u_2v_2 & u_2w_2 & v_2w_2 & 0 & -1 \\ u_1v_3 + u_3v_1 & u_1w_3 + u_3w_1 & v_1w_3 + v_3w_1 & 0 & 0 \\ u_2v_3 + u_3v_2 & u_2w_3 + u_3w_2 & v_2w_3 + v_3w_2 & 0 & 0 \\ u_3v_3 & u_3w_3 & v_3w_3 & 0 & 0 \end{bmatrix}$$

Then $\boldsymbol{\omega}_v$ is calculated as a null vector from

$$\boldsymbol{A}\boldsymbol{\omega}_v = 0. \tag{6.16}$$

Having the coefficients of $\boldsymbol{\omega}_v$, it is easy to calculate $\boldsymbol{K}$ from $\boldsymbol{\omega}_\infty$ via *Cholesky decomposition* [15].

As seen in figure 6.5(b), the calculation of the principal point of the camera is not very accurate, as it is not very close to the centre of the image. There could be two reasons for this: the parallel lines may not have been very accurately estimated, or the calibration patterns in the scene may not have been aligned in a perfectly orthogonal way. As explained and shown in Appendix A.2, the line algorithm by Burns et. al. should be accurate enough. That leaves the latter reason. The next section will outline a method which still makes use of three planes, but which do not have to be orthogonal to each other.

The two constraints arising from the two assumptions of unit aspect ratio and zero skew could have been replaced with two constraints arising from a rectified plane, which is explained in the next section. However, for reasons already pointed out above, this will not make the method any more accurate.

6.4.4 Calibration Method 2

Figure 6.6 shows the same calibration pattern used in the previous section, but this time they are not aligned orthogonally to each other.

In order to obtain five or more constraints from this image, each plane has to be affine rectified. This is done in the following way: for each plane in the image, two vanishing points are calculated and these form the vanishing line $\boldsymbol{l}_\infty$. Normalising $\boldsymbol{l}_\infty$ and using matrix $\boldsymbol{P}$ of equation

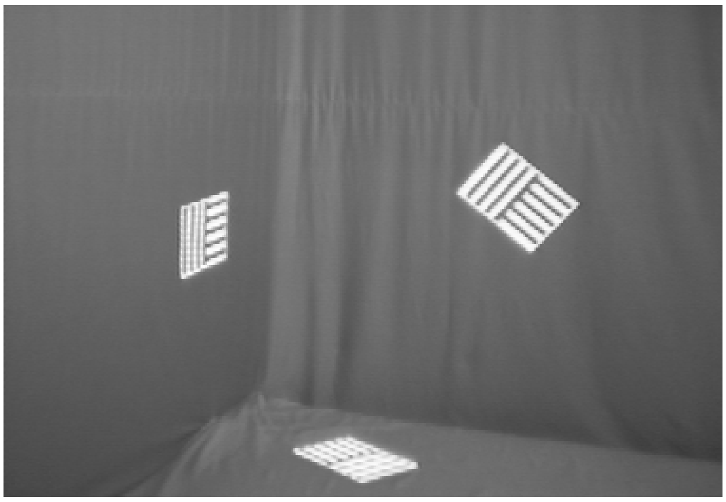


Figure 6.6: Image illustrating three planes in the scene.

(6.10), it is possible to affine rectify each plane in the image, as seen in figure 6.7 for one plane. This makes it possible to calculate, for example, length ratios on parallel line segments in the image. Interpolation methods can be employed to fill in the gaps in the rectification process.

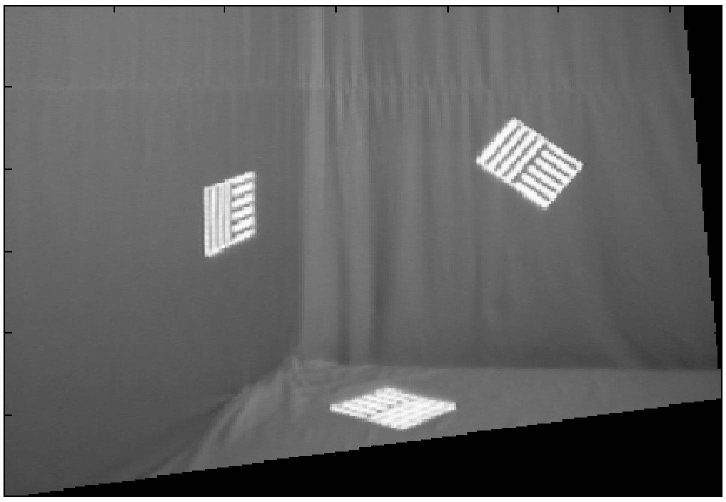


Figure 6.7: Back wall (plane) affine rectified.

The task now is to calculate from the affine rectified image the α and β values, which define the two circular points. Three methods are outlined which provide constraints to calculate α and β [28]. These constraints are obtained as a circle:

Known Angle: If angle θ in the scene between two lines imaged as $\boldsymbol{l}_a$ and $\boldsymbol{l}_b$ (lines are homogeneous vectors) is known, then α and β lie on the circle with centre and radius:

$$(c_\alpha, c_\beta) = \left(\frac{(a+b)}{2}, \frac{(a-b)}{2} \cot\theta \right)$$
$$r = \left| \frac{(a-b)}{2\sin\theta} \right|$$

where $a = -l_{a2}/l_{a1}$ and $b = -l_{b2}/l_{b1}$ are the line directions. If $\theta = \pi/2$, then the circle lies on the α axis.

Equal (unknown) Angles: Knowing that an angle in the scene between two imaged lines with directions a_1 and b_1 is the same as that between two lines imaged with directions a_2 and b_2, then α and β lie on the circle with centre and radius:

$$(c_\alpha, c_\beta) = \left(\frac{(a_1 b_2 - b_1 a_2)}{a_1 - b_1 - a_2 + b_2}, 0 \right)$$
$$r^2 = \left(\frac{a_1 b_2 - b_1 a_2}{a_1 - b_1 - a_2 + b_2} \right)^2 + \frac{(a_1 - b_1)(a_1 b_1 - a_2 b_2)}{a_1 - b_1 - a_2 + b_2} - a_1 b_1.$$

Known Length Ratio: Knowing the length ratio s of two non-parallel line segments in the scene, then figure 6.8 shows the imaged line segments with known endpoints. Writing

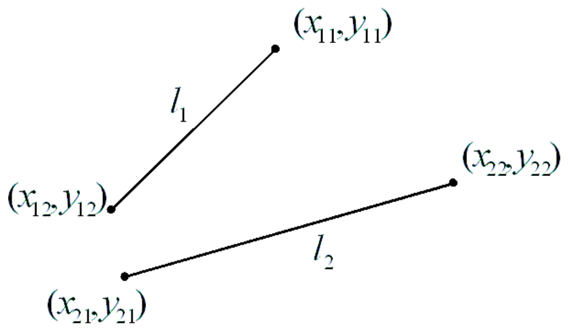


Figure 6.8: Line Ratio Constraint [28].

Δx_n for $x_{n1} - x_{n2}$ and Δy_n for $y_{n1} - y_{n2}$, then α and β lie on the circle with centre and radius:

$$(c_\alpha, c_\beta) = \left(\frac{\Delta x_1 \Delta y_1 - s^2 \Delta x_2 \Delta y_2}{\Delta y_1^2 - s^2 \Delta y_2^2}, 0 \right)$$
$$r = \left| \frac{s(\Delta x_2 \Delta y_1 - \Delta x_1 \Delta y_2)}{\Delta y_1^2 - s^2 \Delta y_2^2} \right|.$$

Two independent constraints are required to solve for α and β, as seen in figure 6.9. If all constraint circles have centres on the same axis, then only intersections in the upper half plane need to be considered.

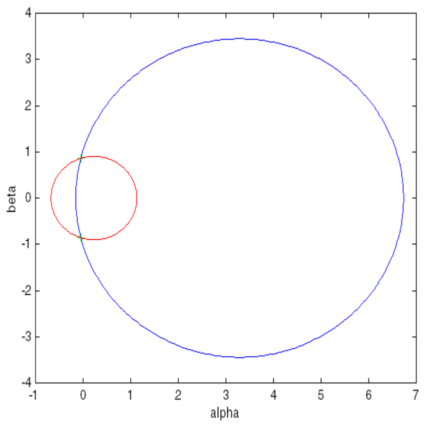


Figure 6.9: Constraint Circles.

In figure 6.6, the ratio of lines and also the angle of each corner of the A4 paper on each plane are known. These can be taken into account in the above equations.

Then it is possible to calculate the circular points as in equation (6.9). Because $\boldsymbol{I}$ and $\boldsymbol{J}$ contain the same information, one takes the real and imaginary parts of either of them to obtain two constraints on $\boldsymbol{\omega}_\infty$: For $\boldsymbol{I}$,

$$\boldsymbol{I}^T \boldsymbol{\omega}_\infty \boldsymbol{I} = 0,$$

and the real and imaginary parts are:

$$(\beta^2 - \alpha^2)\omega_1 - 2\alpha\omega_2 - \omega_3 + 2(l_1(\alpha^2 - \beta^2) + \alpha l_2)\omega_4 + 2(\alpha l_1 + l_2)\omega_5 + (l_1^2\beta^2 - (\alpha l_1 + l_2)^2)\omega_6 = 0$$
$$2\alpha\beta\omega_1 + 2\beta\omega_2 - 2(\beta l_2 + 2\alpha\beta l_1)\omega_4 - 2\beta l_1\omega_5 + 2(\alpha\beta l_1^2 + \beta l_1 l_2)\omega_6 = 0$$

It can be seen then that each rectified plane provides two constraints on $\boldsymbol{\omega}_\infty$. These constraints can be combined as in section 6.4.3.

6.4.5 Conclusion

The two calibration methods described are well suited to images where the origin is not known. But for normal applications these methods are quite complex. Rectifying each plane in the image poses some problems, as it can happen that a vanishing line lies between the origin and the plane to be rectified, and this will warp points closest to the vanishing line to infinity. To compensate for this, the image needs to be translated before rectification.

The rectification process also sometimes creates very large images, which are difficult to work with.

The whole calibration process is also difficult to automate, as the user generally has to select the correct parallel lines or select some features on each plane to obtain the necessary constraints.

6.5 Calibration using a Planar Pattern

6.5.1 Introduction

This section describes an implementation of the camera calibration toolbox similar to that developed by Bouguet[1]. He bases his calibration technique on papers by Zhang [51, 53] and the internal camera model on a paper by Heikkilä and Silvén [22]. In the implementation presented, however, the internal camera model is entirely based on Zhang's technique [51, 53], which is identical to the camera calibration matrix of equation (3.4).

6.5.2 Homography between the Planar Object and its Image

Figure 6.10 shows a square planar calibration pattern consisting of 7×7 blocks of known length of $25mm$.

To establish the homography between the planar object and its image, it can be assumed that the planar object lies at $Z = 0$ in the world coordinate system. Making use of equations (3.1) and (3.8), and representing the i^{th} column vector of the rotation matrix $\boldsymbol{R}$ by $\boldsymbol{r}_i$, the following

[1]The Camera Calibration Toolbox for $Matlab^{®}$ by Jean-Yves Bouguet can be downloaded from his homepage at Caltech: $http://www.vision.caltech.edu/bouguetj/calib\_doc/index.html$ (accessed November 2000). The C implementation of this toolbox is included in the free *Open Source Computer Vision Library* (©2000 Intel Corporation) at http://www.intel.com/research/mrl/research/cvlib/ (accessed November 2000).

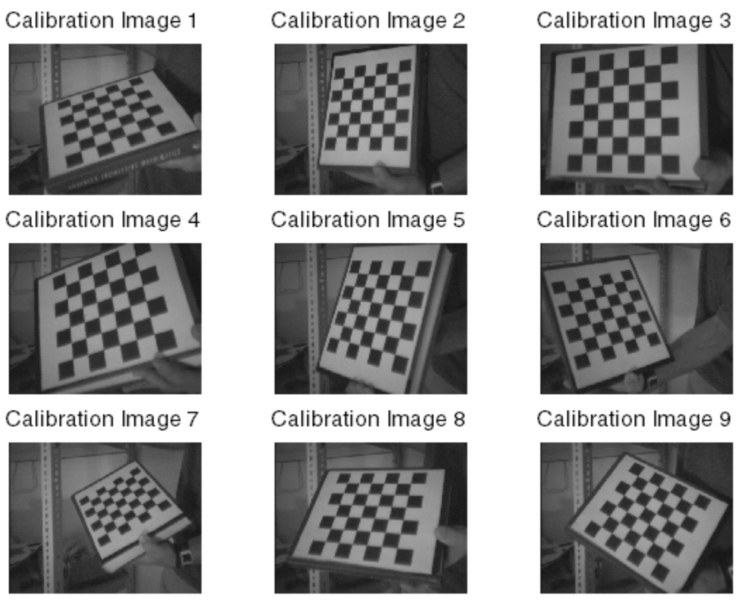


Figure 6.10: Planar Calibration Patterns.

equation is obtained:

$$
\begin{aligned}
s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} &= \boldsymbol{K}\begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 & \boldsymbol{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \\
&= \boldsymbol{K}\begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix}\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}.
\end{aligned}
$$

In this case, because $Z = 0$, the homogenous coordinates of point $\boldsymbol{M}$ are written as $\tilde{\boldsymbol{M}} = [X, Y, 1]^T$. Then a planar object point $\boldsymbol{M}$ is related to its image point $\boldsymbol{m}$ by a 3×3 homography matrix $\boldsymbol{H}$:

$$
s\tilde{\boldsymbol{m}} = \boldsymbol{H}\tilde{\boldsymbol{M}}, \tag{6.17}
$$

with $\boldsymbol{H} = \boldsymbol{K}\begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix}$.

The homography for each image in figure 6.10 is estimated by selecting the 4 corners of the imaged planar object and refining these 4 corners to subpixel accuracy using the method outlined in appendix A.1.3. The world coordinate points $\boldsymbol{M}_{1...4}$ are defined as in figure 6.11.

Zhang [51, 53] uses a maximum likelihood criterion to estimate the homography. As the

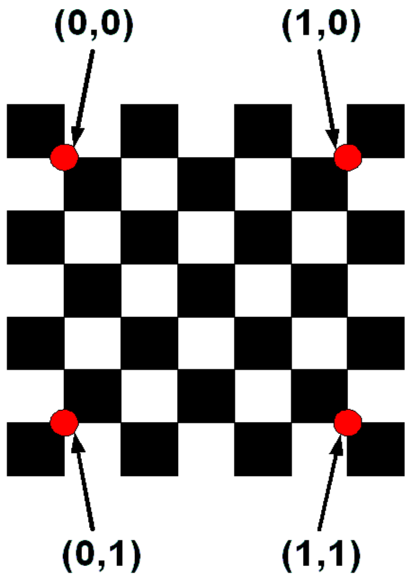


Figure 6.11: World Coordinate Points of Planar Pattern.

image points $\boldsymbol{m}_i$ are corrupted by noise, a maximum likelihood criterion of $\boldsymbol{H}$ is obtained by minimising the following:

$$\min_{\boldsymbol{H}} \sum_i \|\boldsymbol{m}_i - \hat{\boldsymbol{m}}_i\|^2,$$

where

$$\hat{\boldsymbol{m}}_i = \frac{1}{\boldsymbol{h}_3^T \boldsymbol{M}_i} \begin{bmatrix} \boldsymbol{h}_1^T \boldsymbol{M}_i \\ \boldsymbol{h}_2^T \boldsymbol{M}_i \end{bmatrix}$$

with $\boldsymbol{h}_i$ the i^{th} row of $\boldsymbol{H}$. This nonlinear minimisation can be solved using the Levenberg-Marquardt algorithm described in appendix C.

With $\boldsymbol{x} = [\boldsymbol{h}_1^T, \boldsymbol{h}_2^T, \boldsymbol{h}_3^T]^T$, equation (6.17) can be rewritten as follows:

$$\begin{bmatrix} \tilde{\boldsymbol{M}}^T & \boldsymbol{0}^T & -u\tilde{\boldsymbol{M}}^T \\ \boldsymbol{0}^T & \tilde{\boldsymbol{M}}^T & -v\tilde{\boldsymbol{M}}^T \end{bmatrix} \boldsymbol{x} = 0. \tag{6.18}$$

For n points, n above equations are obtained and can be written as a matrix $\boldsymbol{L}\boldsymbol{x} = 0$, where $\boldsymbol{L}$ is a $2n \times 9$ matrix. The solution is then defined as the eigenvector of $\boldsymbol{L}^T\boldsymbol{L}$ associated with the smallest eigenvalue. It should be noted that better results can be achieved by normalising the image points as described in section 4.2.

Once the homography for the 4 corners of each planar pattern has been estimated, it is possible to extract all the remaining corners on the grid, as the number of blocks for each side and the length of each block is known. It is then possible to refine the homography considering all the corners of the grid.

6.5.3 Calculating the Camera Calibration Matrix

For each image, a homography can be estimated as described in the previous section. Writing $\boldsymbol{H} = [\ \boldsymbol{h}_1 \quad \boldsymbol{h}_2 \quad \boldsymbol{h}_3\ ]$, equation (6.17) is rewritten as:

$$\begin{bmatrix} \boldsymbol{h}_1 & \boldsymbol{h}_2 & \boldsymbol{h}_3 \end{bmatrix} = \lambda \boldsymbol{K} \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{t} \end{bmatrix},$$

where λ is a scalar. Because vectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ are orthonormal (a fundamental property of rotation matrices), the following two equations are obtained and give two constraints on the internal parameters of the camera:

$$\boldsymbol{h}_1^T \boldsymbol{K}^{-T} \boldsymbol{K}^{-1} \boldsymbol{h}_2 = 0 \tag{6.19}$$

$$\boldsymbol{h}_1^T \boldsymbol{K}^{-T} \boldsymbol{K}^{-1} \boldsymbol{h}_1 = \boldsymbol{h}_2^T \boldsymbol{K}^{-T} \boldsymbol{K}^{-1} \boldsymbol{h}_2. \tag{6.20}$$

It can be seen that the term $\boldsymbol{K}^{-T}\boldsymbol{K}^{-1}$ represents the image of the absolute conic $\boldsymbol{\omega}_\infty$, as described in section 6.3.

Expanding equation (6.5) of the absolute conic $\boldsymbol{\omega}_\infty$, a symmetric matrix is obtained:

$$\begin{aligned} \boldsymbol{\omega}_\infty = \boldsymbol{K}^{-T}\boldsymbol{K}^{-1} &\equiv \begin{bmatrix} \omega_1 & \omega_2 & \omega_4 \\ \omega_2 & \omega_3 & \omega_5 \\ \omega_4 & \omega_5 & \omega_6 \end{bmatrix} \\ &= \begin{bmatrix} \frac{1}{f_u^2} & -\frac{s}{f_u^2 f_v} & \frac{v_0 s - u_0 f_v}{f_u^2 f_v} \\ -\frac{s}{f_u^2 f_v} & \frac{s^2}{f_u^2 f_v^2} + \frac{1}{f_v^2} & -\frac{s(v_0 s - u_0 f_v)}{f_u^2 f_v^2} - \frac{v_0}{f_v^2} \\ \frac{v_0 s - u_0 f_v}{f_u^2 f_v} & -\frac{s(v_0 s - u_0 f_v)}{f_u^2 f_v^2} - \frac{v_0}{f_v^2} & \frac{(v_0 s - u_0 f_v)^2}{f_u^2 f_v^2} + \frac{v_0^2}{f_v^2} + 1 \end{bmatrix}. \end{aligned} \tag{6.21}$$

Defining $\boldsymbol{\omega}_v = (\omega_1, \omega_2, \omega_3, \omega_4, \omega_5, \omega_6)^T$ and denoting the i^{th} column vector of $\boldsymbol{H}$ by $\boldsymbol{h}_i = [h_{i1}, h_{i2}, h_{i3}]^T$, the following equation is derived:

$$\boldsymbol{h}_i^T \boldsymbol{\omega}_\infty \boldsymbol{h}_j = \boldsymbol{v}_{ij}^T \boldsymbol{\omega}_v, \tag{6.22}$$

where

$$\boldsymbol{v}_{ij} = [h_{i1}h_{j1}, h_{i1}h_{j2} + h_{i2}h_{j1}, h_{i2}h_{j2}, h_{i3}h_{j1} + h_{i1}h_{j3}, h_{i3}h_{j2} + h_{i2}h_{j3}, h_{i3}h_{j3}]^T.$$

It is then possible to rewrite the two constraint equations (6.19) and (6.20) as 2 homogeneous equations in $\boldsymbol{\omega}_v$:

$$\begin{bmatrix} \boldsymbol{v}_{12}^T \\ (\boldsymbol{v}_{11} - \boldsymbol{v}_{22})^T \end{bmatrix} \boldsymbol{\omega}_v = 0. \tag{6.23}$$

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

$$\boldsymbol{V}\boldsymbol{\omega}_v = 0, \tag{6.24}$$

where $\boldsymbol{V}$ is a $2n \times 6$ matrix. The general solution is then defined as the eigenvector of $\boldsymbol{V}^T\boldsymbol{V}$ associated with the smallest eigenvalue. If only 2 images are present, it is possible to assume that the skew s is equal to zero, which is added as an additional constraint ($[0, 1, 0, 0, 0, 0]\boldsymbol{\omega}_v = 0$) to equation (6.24). If only 1 image is present, then it can be assumed that the principal point (u_0, v_0) is equal to the image centre and $s = 0$.

Matrix $\boldsymbol{\omega}_\infty$ is defined up to a scalar ($\boldsymbol{\omega}_\infty = \lambda \boldsymbol{K}^{-T}\boldsymbol{K}^{-1}$), and it is then possible to extract the internal parameters of the camera, once vector $\boldsymbol{\omega}_v$ is known:

$$\begin{aligned} v_0 &= (\omega_2\omega_4 - \omega_1\omega_5)/(\omega_1\omega_3 - \omega_2^2) \\ \lambda &= \omega_6 - [\omega_4^2 + v_0(\omega_2\omega_4 - \omega_1\omega_5)]/\omega_1 \\ f_u &= \sqrt{\lambda/\omega_1} \\ f_v &= \sqrt{\lambda\omega_1/(\omega_1\omega_3 - \omega_2^2)} \\ s &= -\omega_2 f_u^2 f_v/\lambda \\ u_0 &= s v_0/\lambda - \omega_4 f_u^2/\lambda. \end{aligned}$$

The external parameters for each image can also be calculated from equation (6.17), once the camera calibration matrix $\boldsymbol{K}$ is estimated:

$$\begin{aligned} \boldsymbol{r}_1 &= \lambda \boldsymbol{K}^{-1}\boldsymbol{h}_1 \\ \boldsymbol{r}_2 &= \lambda \boldsymbol{K}^{-1}\boldsymbol{h}_2 \\ \boldsymbol{r}_3 &= \boldsymbol{r}_1 \times \boldsymbol{r}_2 \\ \boldsymbol{t} &= \lambda \boldsymbol{K}^{-1}\boldsymbol{h}_3, \end{aligned}$$

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

Optimisation

The above obtained solutions are used as an initial guess to a nonlinear optimisation routine, which is defined as follows:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \|\boldsymbol{m}_{ij} - \hat{\boldsymbol{m}}(\boldsymbol{K}, \boldsymbol{R}_i, \boldsymbol{t}_i, \boldsymbol{M}_j)\|^2. \tag{6.25}$$

Here point $\hat{\boldsymbol{m}}(\boldsymbol{K}, \boldsymbol{R}_i, \boldsymbol{t}_i, \boldsymbol{M}_j)$ is the projection of point $\boldsymbol{M}_j$ in image i. The above minimisation problem can be solved by the *Levenberg-Marquardt* algorithm [35, 40], described in appendix C.

6.5.4 Results

The images of the planar object shown in figure 6.10 were taken by a $Watec^{®}$ Camera, Model: WAT-202B(PAL) and grabbed by a $Asus^{®}$ AGP-V3800 Ultra framegrabber. The image size is 704 $\times$ 576 pixels and the pixel size is $p_u = 0.00734mm$ and $p_v = 0.006467mm$. Table 6.1 compares the results obtained by the method outlined above to the real internal parameters of the camera (estimated as in section 6.2.4) and to the results obtained from Bouguet's calibration toolbox.

	Real Parameters	Bouguet's Calibration Toolbox	Zhang's Method
u_0	362.4945504	316.96784	361.376773
v_0	288.907144	230.10757	268.982949
f_u	954.6457766	1014.81586	1030.742112
f_v	1083.516314	1110.65364	1125.157407
s	0	0	0

Table 6.1: Comparison of the real and estimated internal parameters of the camera.

As can be seen, the method outlined in this section does not produce very accurate results, nor does the original method by Bouguet. A reason for this could be that radial distortion was not taken into account. Zhang [51, 53] also implements an algorithm which deals with radial distortion in the images, which should have resulted in better estimates.

The calibration patterns used in section 6.2 usually consist of two planes at different depths. The lack of different depths in the above method could also result in inaccurate values.

Chapter 7

Stratified 3D Reconstruction

7.1 Introduction

This chapter outlines the steps involved in obtaining a 3D model of an object in a stereo image pair. As mentioned in chapter 2, it is possible to divide 3D vision into geometry groups. This so-called stratification is used in this chapter to calculate the geometric relationships between structures in the image pair.

As explained later in this chapter, the reconstruction algorithm relies heavily on the parallel lines estimated from objects in the images. In fact, it is necessary to obtain parallel lines on 3 different planes pointing in 3 different directions. As such, the model to be reconstructed has to consist of at least 3 planes with different orientations in space, with each plane providing attributes such as parallel markers or structures. This of course puts a great constraint on the models which can be reconstructed. Essentially, simple geometric models such as a cube or the corner of a room provide the necessary constraints. In order to reconstruct any arbitrary object, it needs to be placed inside a scene providing the above mentioned constraints.

Section 7.2 explains the 3 steps of the reconstruction algorithm. Once a full 3D reconstruction has been obtained, section 7.3 shows how *dense stereo matching* is used to obtain a 3D textured model.

7.2 3D Reconstruction

This section follows the structure of chapter 2 very closely. Three steps are needed to obtain a full metric reconstruction, the first step being a projective reconstruction followed by an affine and metric one.

7.2.1 Projective Reconstruction

For this step, the fundamental matrix $\boldsymbol{F}$ needs to be estimated from corner point matches, as already outlined in section 4. The fundamental matrix then provides the means to compute the two projective camera matrices for both the images.

Let the first camera coincide with the origin of the world coordinate system. The projective camera matrix for the first camera is then defined as follows:

$$\boldsymbol{P}_1 = \begin{bmatrix} \boldsymbol{I}_{3\times 3} & \boldsymbol{0}_3 \end{bmatrix}. \tag{7.1}$$

The second projective camera matrix is chosen such that the epipolar geometry corresponds to the retrieved fundamental matrix [37, 52]. Usually it is defined as follows:

$$\boldsymbol{P}_2 = \begin{bmatrix} \boldsymbol{M} & \sigma \boldsymbol{e}_2 \end{bmatrix}, \tag{7.2}$$

where $\boldsymbol{e}_2$ is the epipole in the second image and $\boldsymbol{M}$ is a factor of the fundamental matrix: $\boldsymbol{F} = [\boldsymbol{e}_2]_x \boldsymbol{M}$, where $[\boldsymbol{e}_2]_x$ is the antisymmetric matrix of epipole $\boldsymbol{e}_2$ as described in equation (2.6). This epipole can be extracted from the fundamental matrix as explained in section 4.3. Variable σ represents the global scale of the reconstruction, and as that scale is not known, it is arbitrarily chosen and set to 1. Matrix $\boldsymbol{M}$ is defined as follows:

$$\boldsymbol{M} = -\frac{1}{\|\boldsymbol{e}_2\|^2}[\boldsymbol{e}_2]_x \boldsymbol{F}.$$

The matrix $\boldsymbol{M}$ is not necessarily unique, because if $\boldsymbol{M}$ is a solution, then $\boldsymbol{M} + \boldsymbol{e}_2 \boldsymbol{v}^T$ is also a solution for any vector $\boldsymbol{v}$ [52].

Some reconstructions may appear distorted, and Pollefeys [37] points out that this happens when the plane at infinity crosses the scene. He suggests estimating vector $\boldsymbol{v}$ in such a way that the plane at infinity does not cross the scene. Pollefeys makes use of oriented projective geometry [26] to remedy this. The examples in this thesis did not have the problem described and therefore vector $\boldsymbol{v} = [1, 1, 1]^T$.

7.2.2 Affine Reconstruction

This step involves finding the plane at infinity. As mentioned in section 2.3.2, to upgrade a specific projective representation to an affine representation, a transformation needs to be applied which brings the plane at infinity to its canonical position. This can be achieved with the matrix $\boldsymbol{T}_{PA}$ defined in equation (2.27). The two affine projection matrices can then be defined as follows [37]:

$$\boldsymbol{P}_{Ai} = \boldsymbol{P}_i \boldsymbol{T}_{PA}^{-1} \quad \text{for} \quad i = 1, 2. \tag{7.3}$$

The plane at infinity can be calculated if three or more points on that plane are known. These points are points at infinity and are projections of vanishing points in space. Vanishing points are the intersections of two or more imaged parallel lines (see appendix B). In order to determine the three vanishing points, Faugeras et. al. [11] state that three non-coplanar directions of parallel lines need to be established.

This can easily be verified: as figure 7.1 shows, it can happen that imaged parallel lines from two different planes but pointing in the same direction intersect in the same vanishing point. In the figure, the green and blue imaged parallel lines, although on different planes, intersect in the same vanishing point.

Another problem observed is the one illustrated in figure 7.2. Although all imaged parallel lines lie on three different planes and point in three different directions, all three vanishing point lie on one line, as the normalised vanishing points show. A plane can only be estimated with two or more lines, thus with only one line the plane at infinity cannot be accurately defined.

A correct estimation of the three vanishing points is shown in figure 7.3, where the normalised vanishing points clearly illustrate that they lie on two lines which define the plane at infinity very accurately.

Lines in the image are found with the algorithm by Burns et. al. [4] outlined in appendix A.2. Parallel lines in three different planes and directions are selected by the user as figure 7.4 shows.

Projecting the three vanishing points into space with the two projection camera matrices of equations (7.1) and (7.2), the points at infinity are estimated. Triangulation (see appendix D) is employed to find the best estimate of each point in space. From the three points at infinity, the plane at infinity is calculated as follows:

$$\boldsymbol{V}_i^T \boldsymbol{\pi}_\infty = 0 \quad \text{for} \quad i = 1 \ldots 3, \tag{7.4}$$

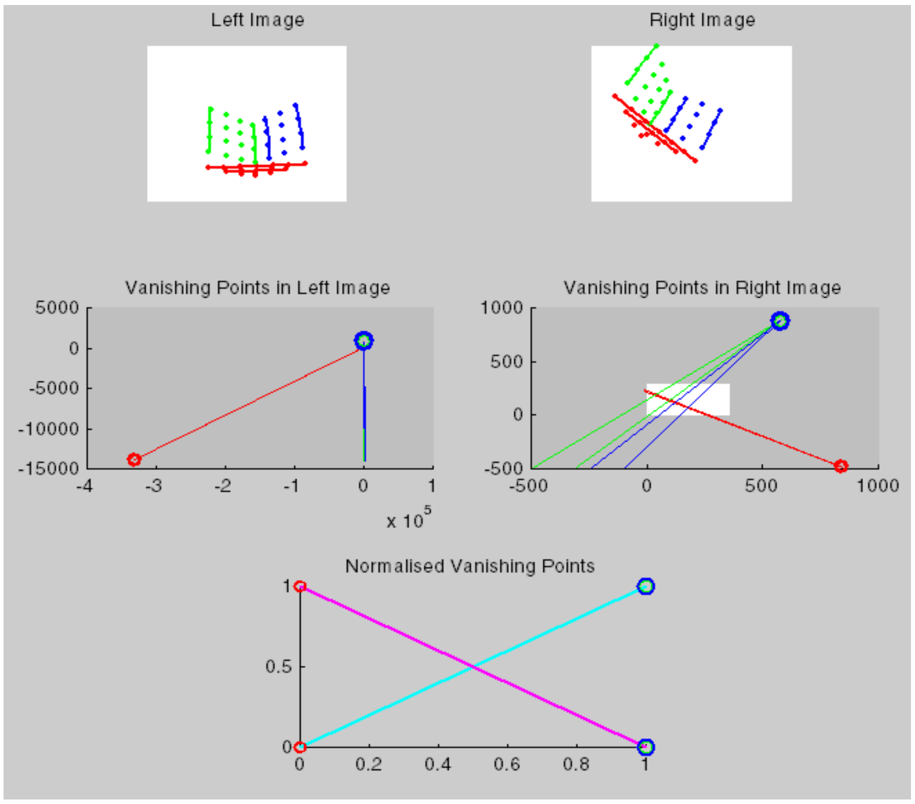


Figure 7.1: Synthetic corner illustrating the vanishing points.

where $\boldsymbol{V}_i$ are the points at infinity. Then $\boldsymbol{\pi}_\infty$ is the non-zero solution of the above linear homogenous system [11].

7.2.3 Metric Reconstruction

After the camera is calibrated and the camera calibration matrix is estimated, it is possible to upgrade the affine representation to a metric representation. The transformation matrix from equation 2.37 achieves this, with matrix $\boldsymbol{B}$ being replaced by the camera calibration matrix $\boldsymbol{K}$.

The two metric projection matrices can then be defined as follows [37]:

$$\boldsymbol{P}_{Mi} = \boldsymbol{P}_{Ai}\boldsymbol{T}_{AM}^{-1} \quad \text{for} \quad i = 1, 2. \tag{7.5}$$

Each individual point is then reconstructed with the help of triangulation methods (see appendix D).

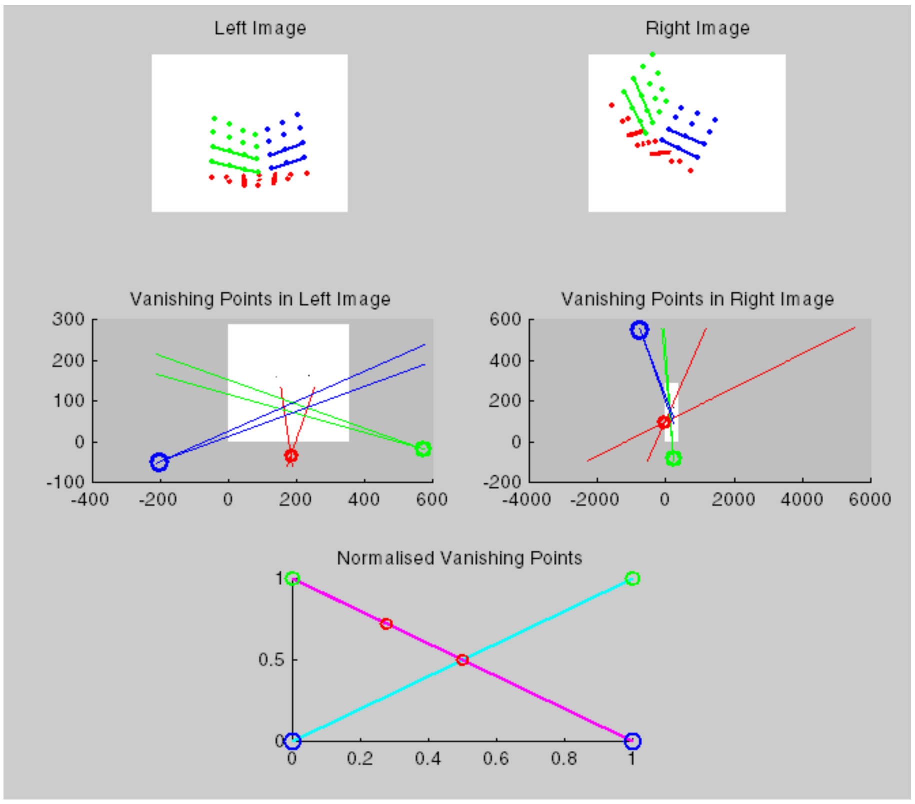


Figure 7.2: Vanishing points defining one line.

7.2.4 Reconstruction Results

For the stereo image pair already shown in figure 7.4, and being only interested in the box in the images, the convex hull for all point matches on the three sides of the box is calculated. (See figure 7.5.)

Reconstructing the points will give a 3D model of the three convex hulls, as seen in figure 7.6. From the 3D model, it is possible to verify that the convex hulls lie at 90° to each other. Table 7.1 shows the results and clearly indicates that the reconstruction has been successful, as the walls of the box are orthogonal.

Angle between Red & Green Plane:	89.98°
Angle between Red & Blue Plane:	90.02°
Angle between Green & Blue Plane:	90.03°

Table 7.1: Angles between the three reconstructed convex hulls.

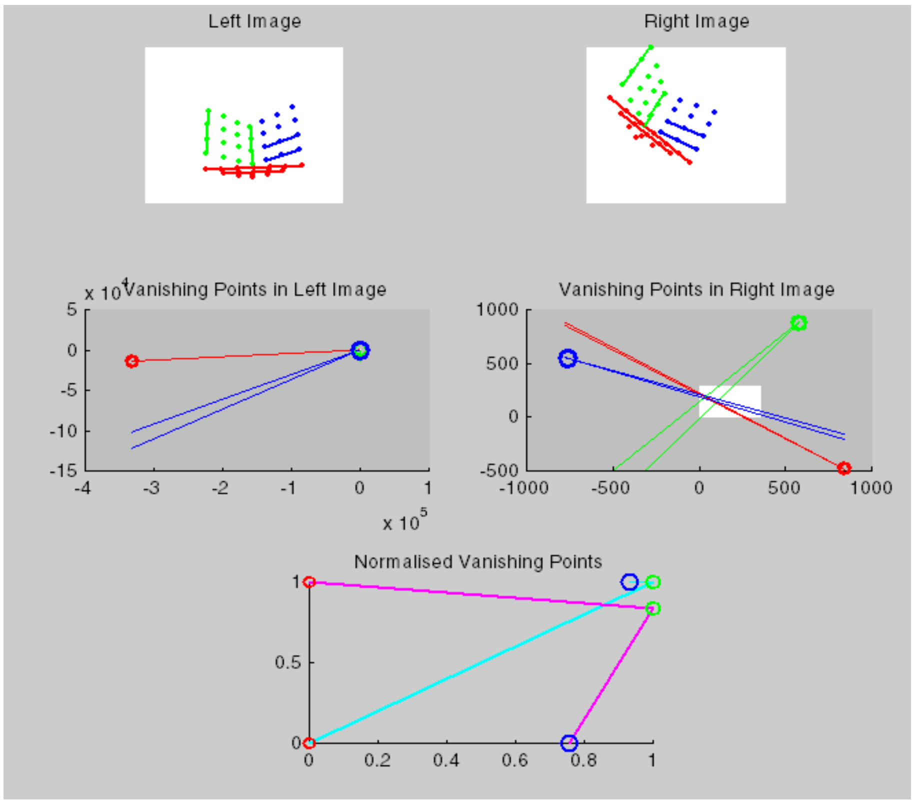
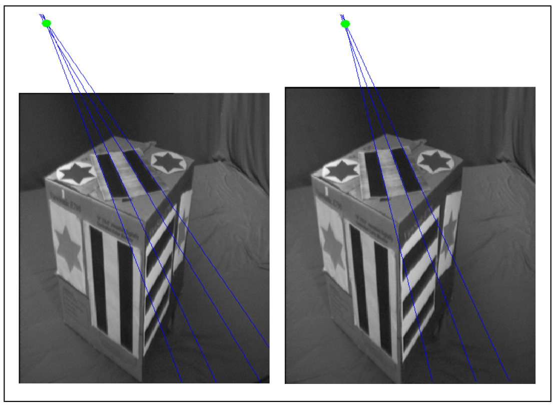


Figure 7.3: Correct estimation of vanishing points.

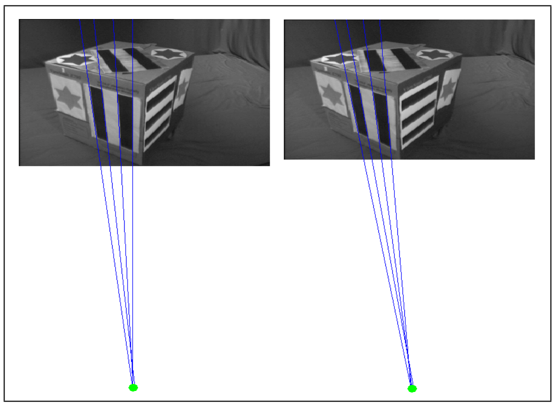
7.3 3D Textured Model

The 3D model of figure 7.6 does not provide a good visualisation of the actual object in the stereo image pair. Figure 7.6 is only suited to verify that the reconstruction was successful. This section will show how to map the texture from the stereo image pair onto the 3D model and in that way provide a better way to visualise the reconstructed object.

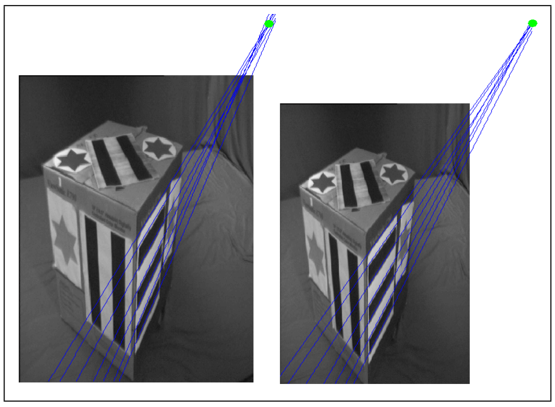
The matching algorithm for uncalibrated stereo images outlined in chapter 5 results in only a few point matches. In order to perform dense stereo matching, the stereo images need to be rectified in a way such that the search space is reduced to one dimension. Section 7.3.1 outlines a rectification method that transforms each image plane such that the epipolar lines are aligned horizontally. Once the images are rectified, it is possible to obtain a match in the second image for nearly every pixel in the first image, as section 7.3.2 explains.



(a)



(b)



(c)

Figure 7.4: Parallel lines estimated in three directions for a stereo image pair.

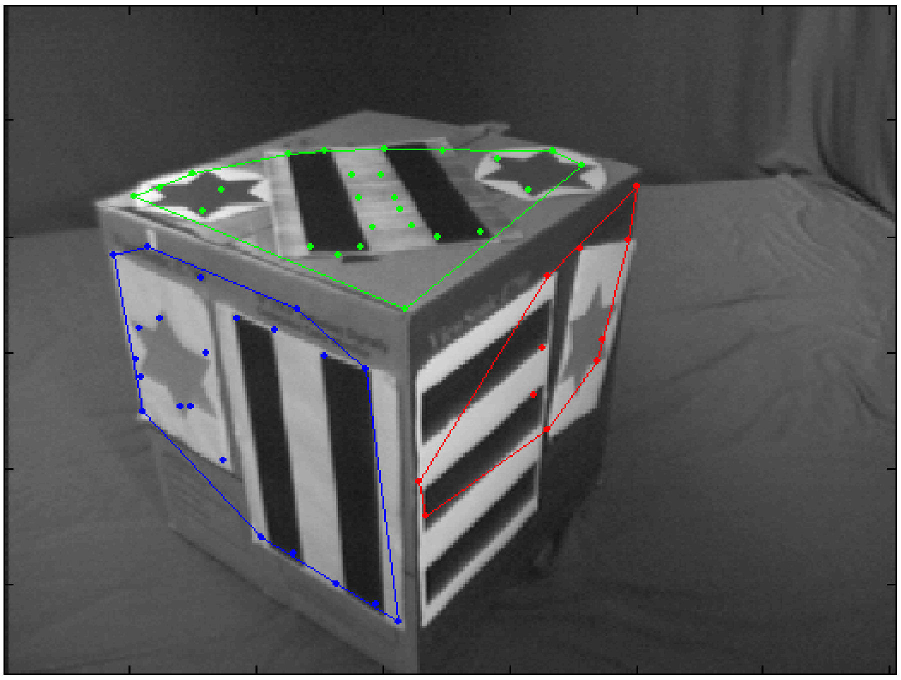


Figure 7.5: Three convex hulls representing the three planes of the box.

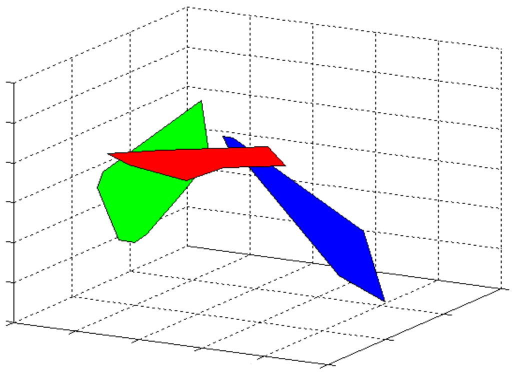
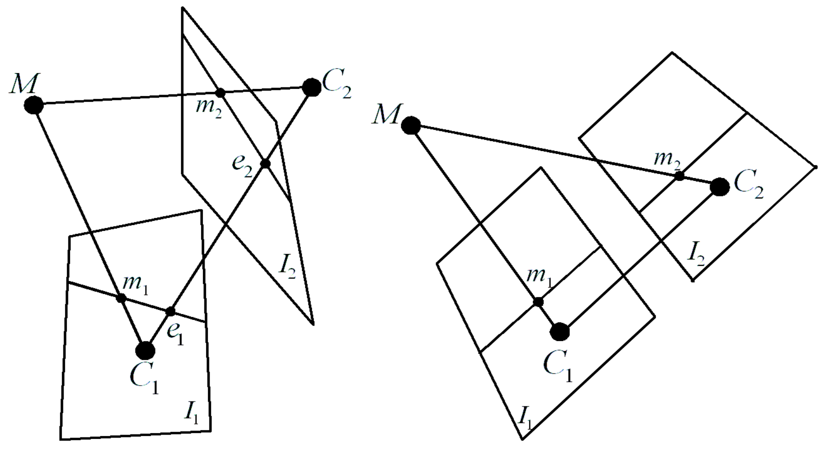


Figure 7.6: Reconstructed convex hulls illustrating the relationship between the three planes of the box.

7.3.1 Rectification of Stereo Images

There are many stereo rectification algorithms which make use of the epipolar constraint to align the images. Pollefeys, Koch and van Gool [39] present a rectification method that can deal with all possible camera geometries with the help of oriented projective geometry [26]. The image is reparameterised with polar coordinates around the epipoles. A somewhat simpler rectification algorithm is presented by Fusiello, Trucco and Verri [13, 14], which only needs the two camera projection matrices to rectify both images. Both algorithms are able to rectify a stereo rig of unconstrained geometry. The latter algorithm was chosen as it did not not employ oriented projective geometry and produced two new rectified camera projection matrices from which a 3D reconstruction is directly possible.



(a) Epipolar Geometry. (b) Rectified Cameras.

Figure 7.7: Rectification Process. Figure obtained from [13, 14].

Figure 7.7 shows the rectification process. In figure 7.7(a) the stereo rig is calibrated and the two camera projection matrices are known (from equation (7.5)). After rectification, two new rectified camera projection matrices are obtained by rotating the old ones around their optical centres until the focal planes become coplanar [13, 14]. The epipoles are now situated at infinity and therefore the epipolar lines are parallel. This introduces a problem when the epipoles before rectification are situated inside or very close to the images, as pixels surrounding the epipoles get mapped to infinity. This results in very large images, which are difficult to operate on.

In order to have horizontal epipolar lines, the baseline $<\boldsymbol{C}_1\boldsymbol{C}_2>$ must be parallel to the new X-axis of both cameras. Conjugate points in both images must have the same vertical coordinate. This can only be achieved if the camera calibration matrix $\boldsymbol{K}$ is the same for both images. That means the focal length is the same and the two image planes are coplanar. (See figure 7.7(b).)

From equation (3.8) and (7.5), it is possible to rewrite $\boldsymbol{P}$:

$$\boldsymbol{P} = [\boldsymbol{Q}|\boldsymbol{q}]. \tag{7.6}$$

Knowing that the focal plane is parallel to the image plane and contains the optical centre $\boldsymbol{C}$, the coordinates $\boldsymbol{c}$ of $\boldsymbol{C}$ are as follows:

$$\boldsymbol{c} = -\boldsymbol{Q}^{-1}\boldsymbol{q}. \tag{7.7}$$

$\boldsymbol{P}$ can then be rewritten:

$$\boldsymbol{P} = [\boldsymbol{Q}| - \boldsymbol{Q}\boldsymbol{c}]. \tag{7.8}$$

The new rectified camera projection matrices are then defined as in equation (7.8):

$$\boldsymbol{P}_{Ri} = \boldsymbol{K}[\boldsymbol{R}| - \boldsymbol{R}\boldsymbol{c}_i] \quad \text{for} \quad i = 1, 2. \tag{7.9}$$

The optical centres are calculated for both cameras from equation (7.7) and the rotation matrix $\boldsymbol{R}$ is the same for both cameras. The row vectors of $\boldsymbol{R}$, $\boldsymbol{r}_{1\ldots3}$, represent the X, Y and Z axes of the camera reference frame in world coordinates.

Fusiello et. al. [13, 14] define then three steps for rectification:

1. The new X-axis parallel to the baseline is $\boldsymbol{r}_1 = (\boldsymbol{c}_1 - \boldsymbol{c}_2)/\|\boldsymbol{c}_1 - \boldsymbol{c}_2\|$.
2. The new Y-axis orthogonal to X and to any arbitrary vector $\boldsymbol{k}$ is $\boldsymbol{r}_2 = \boldsymbol{k}\perp\boldsymbol{r}_1$.
3. The new Z-axis orthogonal to XY is $\boldsymbol{r}_3 = \boldsymbol{r}_1\perp\boldsymbol{r}_2$.

($\boldsymbol{x}\perp\boldsymbol{y}$ describes that vector $\boldsymbol{x}$ is orthogonal to vector $\boldsymbol{y}$.)

The vector $\boldsymbol{k}$ above is set equal to the Z unit vector of the old left matrix, and as such constrains the new Y-axis to be orthogonal to both the new X and old left Z-axis. Fusiello et. al. [13, 14] point out that the algorithm only fails when the optical axis is parallel to the baseline (when there is forward motion).

Writing the old camera projection matrices and the new rectified camera projection matrices as in equation (7.6), the rectifying transformation matrices are as follows:

$$\boldsymbol{T}_i = \boldsymbol{Q}_{Ri}\boldsymbol{Q}_{Mi} \quad \text{for} \quad i = 1, 2. \tag{7.10}$$

This transformation maps the image plane of $\boldsymbol{P}_{M1,M2}$ onto the image plane of $\boldsymbol{P}_{R1,R2}$.

When rectifying an image pair, an arbitrary translation needs to be applied to both images to bring them into a suitable region of the image plane. Figure 7.8 shows the result of rectifying the stereo image pair of the previous section. The epipolar lines for the points in the first image are displayed in the second image. Note that due to calibration inaccuracies some of the points in the second image do not lie on their corresponding epipolar lines.

Fusiello et. al. [13, 14] show in their paper that reconstructing directly from the rectified images does not introduce any major errors and compares favourably with the reconstruction from the original images.

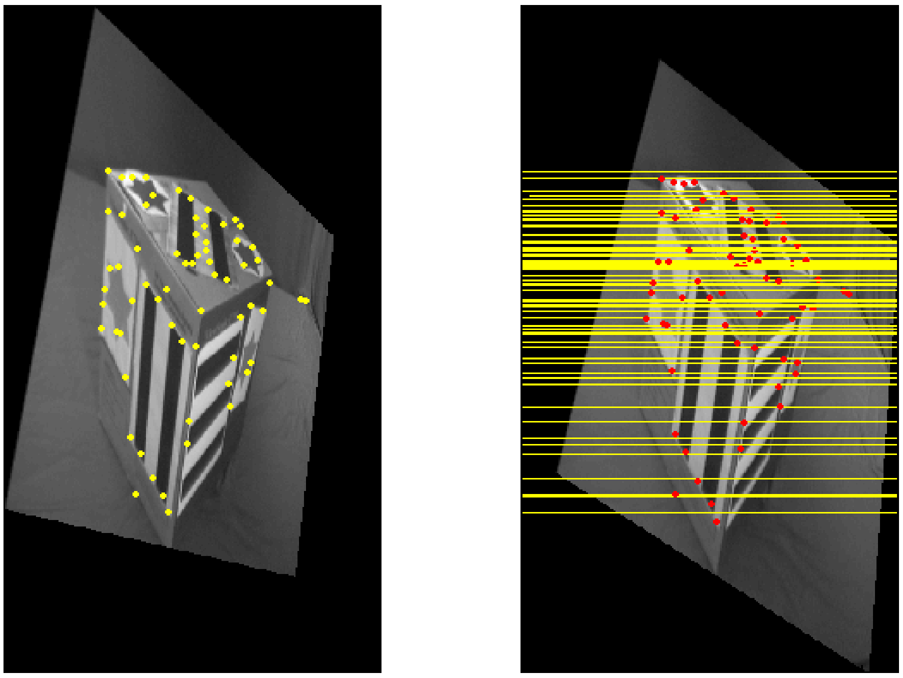


Figure 7.8: Rectified stereo image pair with horizontal epipolar lines.

7.3.2 Dense Stereo Matching

Many different problems arise when attempting dense stereo matching. The most notable problem is occlusions in the images, which simply means that points in one image have no corresponding point in the other one. Ambiguity is a problem when a point in one image can correspond to more than one point in the other image. The intensity can vary from one image to the other and makes the same point in both images look different. Fusiello, Roberto and Trucco [12] have developed a robust area-based algorithm, which addresses all these problems.

They assume that the intensity stays the same for each point in both images and then calculate similarity scores. As the images are rectified, the search space is reduced to the corresponding horizontal line in the right image of each pixel in the left image. Then a small window placed on each pixel in the left image is compared to a window placed in the right image along the corresponding horizontal line. The similarity measure employed is the normalised *sum of squared differences* or SSD error:

$$C(x,y,d) = \frac{\sum_{i=-n}^{n} \sum_{j=-m}^{m} [I_l(x+i,y+j) - I_r(x+i+d,y+j)]^2}{\sqrt{\sum_{i=-n}^{n} \sum_{j=-m}^{m} I_l(x+i,y+j)^2 \sum_{i=-n}^{n} \sum_{j=-m}^{m} I_r(x+i+d,y+j)^2}}. \quad (7.11)$$

To obtain the disparity for a pixel in the left image, the SSD error is minimsed:

$$d_c(x,y) = \arg\min_d C(x,y,d). \quad (7.12)$$

In order to obtain subpixel precision, a curve is fitted to the errors in the neighbourhood of the minimum:

$$\boldsymbol{D}_{x,y} = d_c + \frac{1}{2} \frac{C(x,y,d_c-1) - C(x,y,d_c+1)}{C(x,y,d_c-1) - 2C(x,y,d_c) + C(x,y,d_c+1)}. \quad (7.13)$$

Multiple correlation windows are used to find the smallest SSD error. For each pixel in the left image, the correlation is performed with nine different windows, as figure 7.9 shows. The disparity with the smallest SSD error value is retained. As Fusiello et. al. [12] point out, a window yielding a smaller SSD error is more likely to cover a constant depth region.

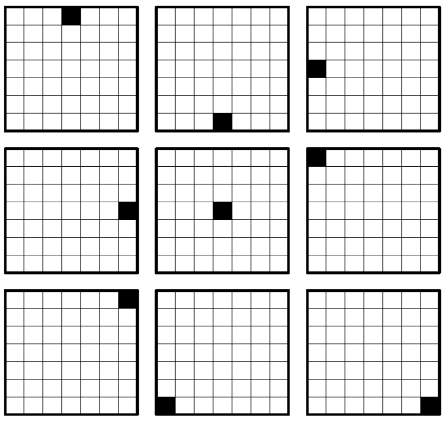
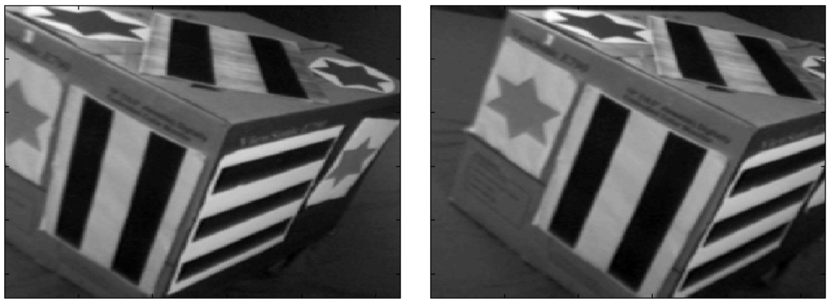


Figure 7.9: Nine different correlation windows. The pixel for which disparity is computed is highlighted. Figure obtained from [12].

Occlusions are detected by reversing the images and then rerunning the algorithm. This process is called *left-right consistency*. For each point in the left image the disparity $d_l(x)$ is computed. Then the images are reversed and the algorithm is rerun. If $d_l(x) = -d_r(x + d_l(x))$, then the point will keep the left disparity. Otherwise the point is marked as occluded. Fusiello et. al. [12] assume that occluded areas occur between two planes at different depths and as such assign the disparity of the deeper plane to the occluded pixel.

From the two rectified images in figure 7.8, the common region of both images is selected (figure 7.10(a)). Then from the few point matches used in the reconstruction algorithm, the minimum and maximum disparity values are calculated to limit the horizontal search space. Figure 7.10(b) shows the resultant disparity map.


(a) Epipolar Geometry.

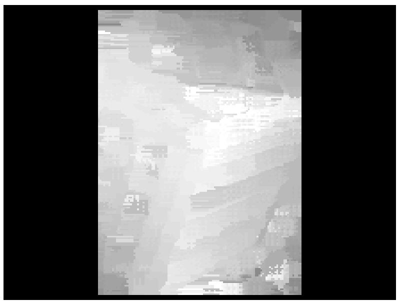

(b) Rectified Cameras.

Figure 7.10: Disparity map calculated on stereo image pair.

If the maximum disparity value is relatively large, then the disparity map is small. As figure 7.10(b) shows, both sides of the disparity map are cropped. Due to the rectification process it is sometimes possible that the disparity is larger than the size of the common region, in which case no texture map can be obtained.

7.3.3 Results

The 3D textured model is obtained from each pixel of the disparity map. The index of the disparity map serves as an index into the left image and the disparity as the horizontal offset of the index in the right image. With the help of the rectified camera projection matrices of equation (7.9) and the triangulation method outlined in appendix D, each point is reconstructed and assigned with the average pixel value from both images. The final reconstructed model is seen in figure 7.11. Better results could be achieved by interpolation techniques.

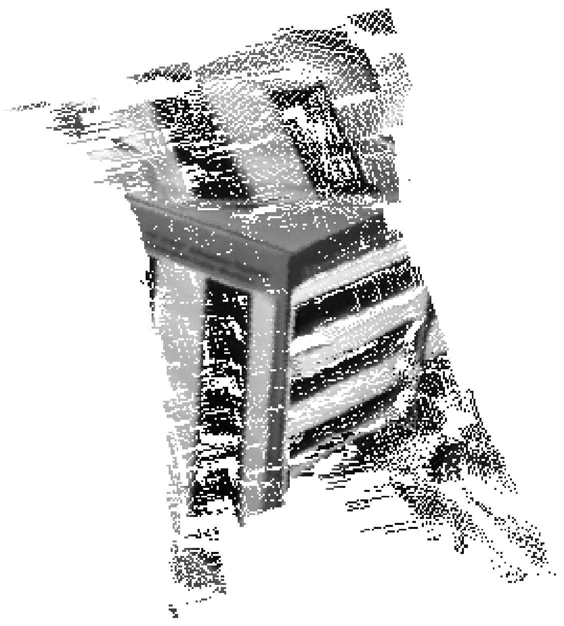
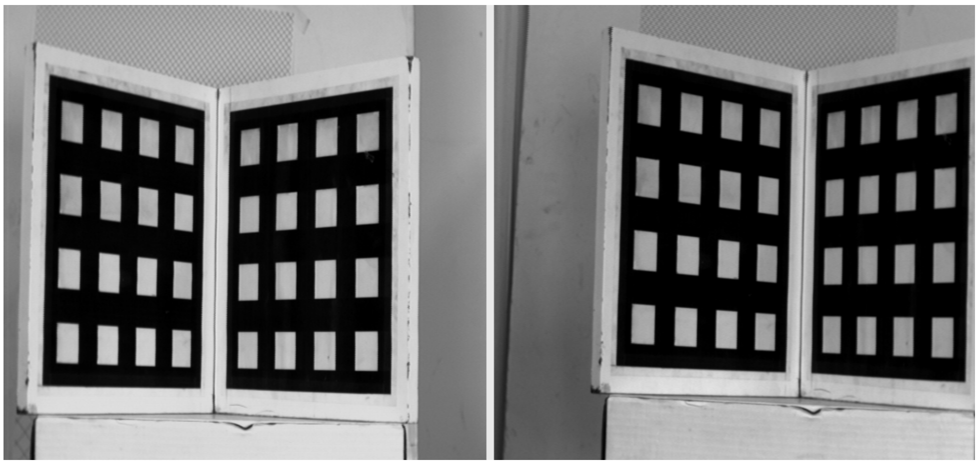


Figure 7.11: Reconstructed 3D texture model.

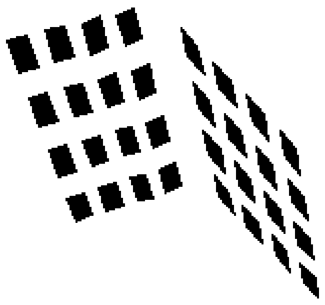
7.4 Other Reconstruction Results

A texture map is not always necessary for visualisation. Figure 7.12(b) and (c) shows the reconstruction of the squares of two calibration panels[1] at 90° to each other of figure 7.12(a). The angle between the two planes in the images was calculated to be 89.26° after reconstruction.

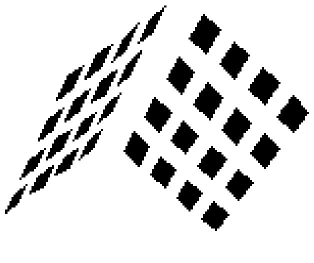
[1]Stereo images, point matches for both images, and the calibration matrix were obtained from the *INRIA-Robotvis project*.



(a) Stereo Image Pair.



(b) Reconstruction View 1.



(c) Reconstruction View 2.

Figure 7.12: Simple Reconstruction of a calibration pattern.

7.5 Conclusion

There are various reasons why the reconstruction and rectification algorithms may fail. As pointed out in section 7.2.1, it is possible to obtain a distorted reconstruction when the plane at infinity crosses the scene. This can only be solved with the help of oriented projective geometry, which is not considered in this thesis.

Obtaining a texture map also proves to be very difficult. First of all the rectification process can result in very large images due to epipoles lying too close to the image plane. Also, when the disparity values become larger than the rectified images, no texture map can be obtained.

All those reasons made it difficult to obtain more reconstructions. The following chapter will look at ways of solving the above problems.

Chapter 8

Conclusions

The work presented in this thesis deals with the extraction of 3D geometry from stereo images. The problem is decomposed into a number of tasks, each task being associated with a specific geometric group. Existing techniques have been implemented and combined to form a relatively easy algorithm, which is straightforward to use. Minimal user intervention is achieved by automating most of the tasks.

The matching algorithm incorporates the best aspects from two papers. The number of correct initial matches has been improved and the algorithm is also capable of working only on a few features. A disadvantage is the memory intensive evaluation of the correlation-weighted proximity matrix if a few hundred features are detected in the images. A minimum of only eight correspondences are needed for the fundamental matrix estimation, thus only the most prominent corners should be extracted from the images. But with computing power increasing rapidly and the cost of memory decreasing, the matching algorithm can easily operate on a 500×500 proximity matrix.

The only part of the reconstruction algorithm which is not automated is the selection of the imaged parallel lines in both images. It is possible to automatically select parallel lines by detecting dominant line directions in histograms. But it is not possible to distinguish between lines pointing in the same direction and lying on the same plane and lines pointing in the same direction but lying on different planes. The only way to solve this is by making sure that parallel lines in the scenes on different planes point in different directions.

The rectification algorithm implemented in this thesis works very well when the epipoles lie far away from the images. It is impossible to make use of this algorithm if the epipoles lie inside the images, as very large images are obtained. The rectification algorithm by Pollefeys, Koch and van Gool [39] should rather be implemented as the rectified images will always be of a certain manageable size.

The problem with the dense stereo matching algorithm implemented in this thesis is that much of the information on both sides of the rectified images is lost due to large calculated initial disparity values. The only way to obtain a full textured model of the whole scene is by expanding the stereo pair to a image sequence of the same scene. This would not only expand the possibilities of the dense stereo matching algorithm, but it would also enable other algorithms such as the *modulus constraint* [37] to be included in the reconstruction problem. It would be possible to estimate the plane at infinity without having to define parallel lines in the scene.

Camera calibration could also be included directly in the reconstruction problem by estimating *Kruppa's equations*, instead of calibrating the camera separately. This would allow for a system that is totally independent of any user intervention. Another interesting aspect would be that cameras could be changed during a reconstruction task and recalibrate themselves again automatically. But as pointed out in section 6.3.2, much work still needs to be done in that area as the whole process of solving for *Kruppa's equations* is very complex and computationally expensive.

An advantage of breaking up the reconstruction problem into different tasks is that it makes it possible to exchange algorithms for each part at a later stage. This is especially important when developing this system further, without having to redefine a new approach.

A limitation of this thesis is that only rigid scenes can be considered in the reconstruction. Moving objects in a scene would cause the reconstruction algorithm to fail. Possibly most of the research in computer vision will be geared towards this area. An important application must surely be the tracking of people inside a scene and obtaining a 3D model at the same time.

Appendix A

Feature Extraction

A.1 Corner Detection

A definition of a corner is "*features formed at boundaries between only two image brightness regions, where the boundary curvature is sufficiently high*" [45]. One way of finding corners in an image is by first segmenting the image in some way to find the shape of an object and then using an edge detector to find the edge chains. Searching the edges for turnings in the boundary will result in a corner. An example of this would be using the *Hough transform* to find straight lines in an image and then finding the end points of these lines. The problem with these techniques is that they rely on prior segmentation of the shapes in the images and inaccuracies will appear due to the segmentation. It is therefore important to find a corner detector that operates on the image directly.

The two corner detection algorithms described here are the *Kitchen and Rosenfeld* [23] and *Harris-Plessey* [16] corner detectors. The latter one can also be used to refine the corners up to subpixel accuracy.

A.1.1 Kitchen and Rosenfeld Corner Detector

The *Kitchen and Rosenfeld* corner finder [23] is used to extract initial corners from the image only up to pixel accuracy. This algorithm applies an edge detector to the gray level image and finds changes in direction along the edges. These changes should correspond to turns in the boundary of an object. This can be achieved by finding the gradient direction in the image, which is given by

$$\tan(\theta(x,y)) = \frac{I_y(x,y)}{I_x(x,y)}, \tag{A.1}$$

where I_x and I_y are the first partial derivatives of the image I at point (x, y). The derivatives can be calculated using horizontal and vertical *Sobel* operators to measure the x and y components of the gradient in the image. The *Sobel* operator is then applied to the gradient direction image to find the changes in direction of the edge. This is the curvature measure, which is multiplied by the gradient magnitude from the first image to obtain the corner magnitude measure.

Calculating the partial derivatives of θ (here the arguments of functions are omitted for clarity):

$$\theta_x = \frac{I_{xy}I_x - I_{xx}I_y}{I_x^2 + I_y^2}$$
$$\theta_y = \frac{I_{yy}I_x - I_{xy}I_y}{I_x^2 + I_y^2},$$

the corner magnitude is defined as follows:

$$\kappa = \frac{I_x\theta_y - I_y\theta_x}{I_x^2 + I_y^2} = \frac{I_{xx}I_y^2 + I_{yy}I_x^2 - 2I_{xy}I_xI_y}{I_x^2 + I_y^2}.$$

This expression finds the rate of change of gradient direction along the edge in the image, multiplied by the gradient magnitude. It can be regarded as the curvature of a contour line. Thresholding this image will result in points with high curvature being selected. The corners lie along the strongest edges of the image.

A.1.2 Harris-Plessey Corner Detector

The *Harris* or *Plessey* corner finder [16] also finds corners by considering points of high curvature. It is based on the *Moravec* corner detector [34].

The Moravec corner detector starts by looking for a large variation in intensity in some local window in the image in every direction. This can be expressed mathematically by an autocorrelation function, which, when thresholded, yields the desired corner points. The result was found to be very noisy due to the rectangular nature of the window, and that the operator is very sensitive to strong edges. Harris and Stephens [16] modified the Moravec operator by using a circular Gaussian window.

Their autocorrelation function is as follows:

$$E(x, y) = Ax^2 + 2Bxy + Cy^2, \tag{A.2}$$

where

$$A = I_x^2 \otimes w$$
$$B = I_{xy} \otimes w$$
$$C = I_y^2 \otimes w,$$

$w = e^{-(\text{winx}^2+\text{winy}^2)/2\sigma^2}$ is the smooth Gaussian window and $\otimes$ is the convolution operator. Variables winx and winy define the window size. The autocorrelation function can be rewritten for small shifts:

$$E(x, y) = [x, y]\boldsymbol{G}[x, y]^T, \tag{A.3}$$

where

$$\boldsymbol{G} = \begin{bmatrix} A & B \\ B & C \end{bmatrix} \tag{A.4}$$

is a 2×2 matrix. The following operator is then used to extract corners from the image:

$$R(x, y) = \det(\boldsymbol{G}) - k\,(\text{trace}(\boldsymbol{G}))^2. \tag{A.5}$$

The value for k is usually taken to be 0.04 to take into account high contrast pixel step edges [54]. Thresholding R will then give the desired corners.

A.1.3 Subpixel Corner Detection

To refine the corners estimated by the *Kitchen and Rosenfeld* or *Harris-Plessey* corner detectors to subpixel accuracy, an algorithm is implemented which was developed by Jean-Yves Bouguet and included in the *Open Source Computer Vision Library* (©2000 Intel Corporation)[1] [6]. With reference to figure A.1, it can be observed that every vector from the centre $\boldsymbol{q}$ to a point $\boldsymbol{p}$ in the neighbourhood of $\boldsymbol{q}$ is orthogonal to the image gradient at $\boldsymbol{p}$ and subject to image and measurement noise. Point $\boldsymbol{q}$ is the subpixel accurate corner location or sometimes also called the *radial saddle point*.

[1]The *Open Source Computer Vision Library* (©2000 Intel Corporation) can be downloaded from the official website: http://www.intel.com/research/mrl/research/cvlib/ (accessed November 2000).

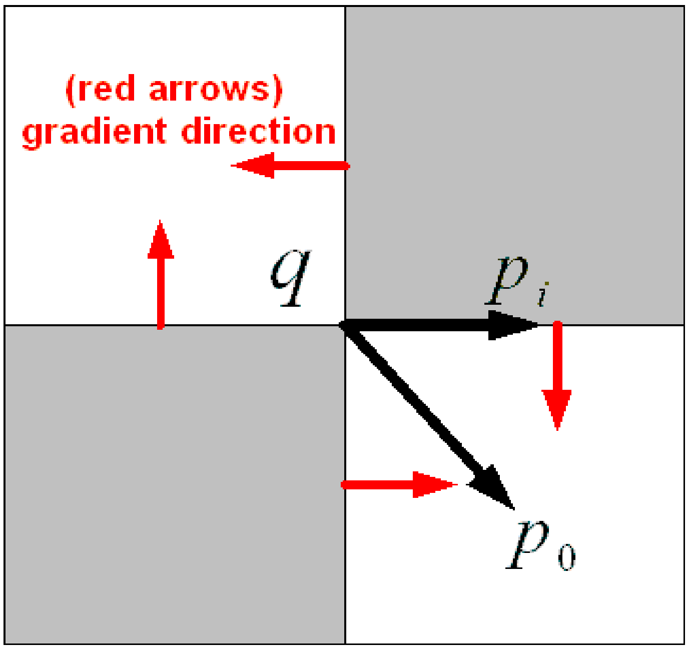


Figure A.1: Illustration of subpixel corner estimation (©2000 Intel Corporation).

The following equation can be defined:

$$\varepsilon_i = \nabla I_{p_i}^T (\boldsymbol{q} - \boldsymbol{p}_i), \tag{A.6}$$

where ∇I_{p_i} is the image gradient at point $\boldsymbol{p}_i$ in the neighbourhood of $\boldsymbol{q}$. To find the best estimate for $\boldsymbol{q}$, ε_i needs to be minimised. Equation A.6 defines a system of equations where ε_i is set to zero:

$$\left(\sum_i \nabla I_{p_i} \nabla I_{p_i}^T\right) \boldsymbol{q} - \left(\sum_i \nabla I_{p_i} \nabla I_{p_i}^T \boldsymbol{p}_i\right) = 0, \tag{A.7}$$

where the gradients are summed up in the neighbourhood of $\boldsymbol{q}$. Using $\boldsymbol{G}$ (defined as in equation (A.4)) for the first gradient term and $\boldsymbol{b} = \boldsymbol{G}\boldsymbol{p}_i$ for the second one, equation (A.7) can be rewritten:

$$\boldsymbol{q} = \boldsymbol{G}^{-1}\boldsymbol{b}. \tag{A.8}$$

Variable $\boldsymbol{q}$ defines a new neighbourhood (window) centre, and the above process is iterated until $\boldsymbol{q}$ does not move more than a certain threshold. This threshold is set to a resolution of 0.05 in the implementation of the *Open Source Computer Vision Library*.

A.2 Extracting Straight Lines

This section explains the straight line algorithm of Burns et. al. [4]. An implementation of this algorithm can be found in the *DARPA Image Understanding Environment* software package (©Amerinex Applied Imaging)[2] [2].

The use of the *Hough transform* was considered for finding straight lines, but it was found that it was not accurate enough. The lines were not perfectly parallel to an edge.

The algorithm described here makes use of the underlying intensity surrounding each edge and uses this information to extract a straight line parallel to the edge. The algorithm is divided into two parts:

1. Grouping pixels into line-support regions.
2. Interpreting the line-support region as a straight line.

A.2.1 Line-support Regions

The image is convolved by a mask (*Sobel mask*) in the x and y direction to find the gradient magnitude and orientation. The gradient orientation is calculated as in equation (A.1) of section A.1 and is then grouped into regions. This is done using fixed and overlapping partitioning: the 360° range of gradient directions are quantised into a small set of regular intervals (8 intervals of 45° each) and each gradient vector is labelled according to the partition into which it falls (see figure A.2).

This means that pixels on a straight line will fall within a certain partition, whereas adjacent pixels that are not part of the same straight line will usually have different orientations and thus fall into a different partition.

A problem with this fixed partioning is "*the arbitrary placement of the boundaries of the partitions and the resulting insensitivity to the possible distributions of edge directions of any particular straight line*" [4]. As an example, a straight line can produce fragmented support regions if the gradient directions lie across a partition boundary. To remedy this, two overlapping sets of partitions are used. This means that for the first partition the first bucket is centered at 0° and for the second partition the first bucket is centered at 22.5°. If one partition fragments one line, the other will place that line entirely within a partition.

[2]For more information and to download the software package go to: http://www.aai.com/AAI/IUE/ (accessed November 2000).

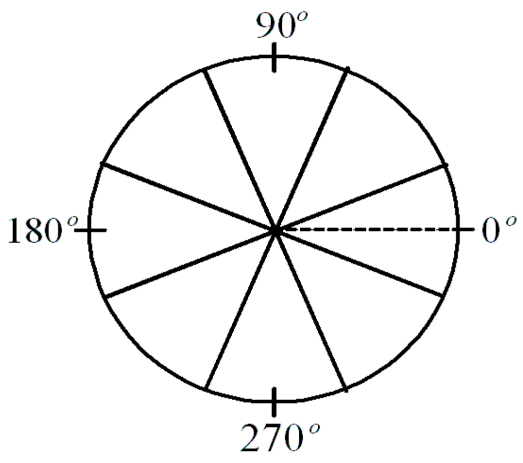


Figure A.2: Gradient Space Partioning. Figure obtained from [4].

These two partition representations have to be merged in such a way that a single line is associated with only one line-support region. This is done as follows:

- Line lengths are determined for every region.
- Each pixel is a member of two regions, so every pixel will vote for and is associated with that region of the two that provides the longest interpretation (or has the biggest area).
- Each region receives a count of the number of pixels that voted for it.
- The support of each region is given as the percentage of the total number of pixels voting for it.

The regions are then selected if the support is greater than 50%.

A.2.2 Interpreting the Line-Support Region as a Straight Line

Each line-support region represents a candidate area for a straight line as the local gradient estimates share a common orientation. The intensity surface associated with each region is then modelled by a planar surface. The parameters of the plane are found by a weighted least-squares fit to the surface. The equation of the plane is as follows: $z(x, y) = Ax + By + C$.

The line must then lie perpendicular to the gradient of the fitted plane. To find this line, the fitted plane is intersected with a horizontal plane at a height of the average weighted intensity (see figure A.3).

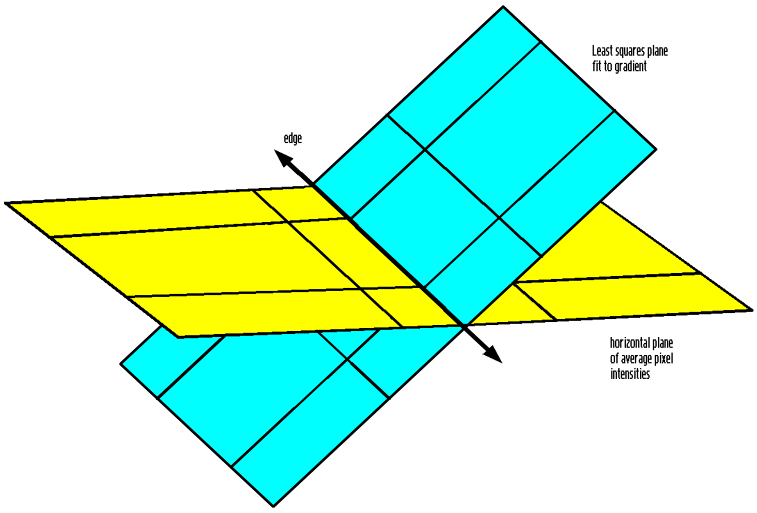


Figure A.3: Two planes intersecting in a straight line. Figure obtained from [4].

The intersection is a straight line, which is parallel to the edge in the image. Once all these lines have been found, only lines of a certain length or orientation are retained.

A comparison with the Hough transform is shown in figure A.4. The Hough transform is clearly not as accurate as the algorithm outlined here. The reason for this is that the Hough transform only searches for an edge and fits a line to it, while with the Burns line algorithm, the whole area surrounding an edge is taken into account when fitting a line.

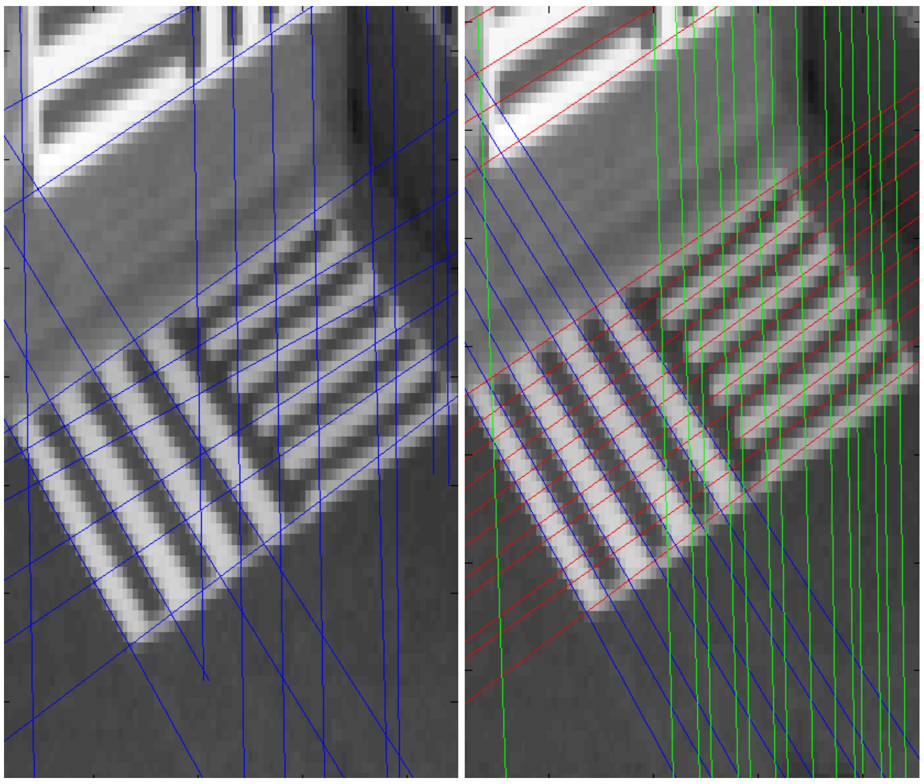

(a) Hough Transform. (b) Burns Line Algorithm.

Figure A.4: Comparison between the Hough Transform and Burns Line Algorithm.

Appendix B

Vanishing Point Estimation

A vanishing point is the intersection of two or more imaged parallel lines. For two lines $\boldsymbol{l}_1$ and $\boldsymbol{l}_2$, the intersection is simply the cross product $\boldsymbol{x} = \boldsymbol{l}_1 \times \boldsymbol{l}_2$. For additional lines it becomes more difficult to solve for the vanishing point.

Because of measurement error of the lines in the image, these lines will not generally intersect in a unique point. Liebowitz et. al. [28, 29] implement a *maximum likelihood estimate* or MLE to find the best estimate of the vanishing point. This is done "*by computing a set of lines that do intersect in a single point, and which minimise the sum of squared orthogonal distances from the endpoints of the measured line segments*" [28].

If the endpoints of the measured line $\boldsymbol{l}$ are $\boldsymbol{x}_a$ and $\boldsymbol{x}_b$ (as in figure B.1), then the MLE minimises the following quality:

$$C = \sum_i d^2(\hat{\boldsymbol{l}}_i, \boldsymbol{x}_{a_i}) + d^2(\hat{\boldsymbol{l}}_i, \boldsymbol{x}_{b_i})$$

subject to the constraint $\boldsymbol{v}^T \hat{\boldsymbol{l}}_i = 0$, where $d(\boldsymbol{x}, \boldsymbol{l})$ is the perpendicular image distance between point $\boldsymbol{x}$ and line $\boldsymbol{l}$.

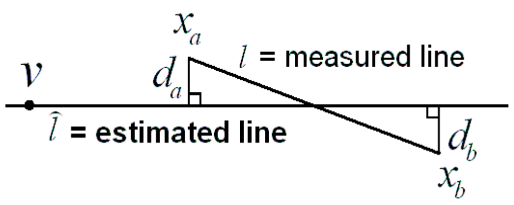


Figure B.1: Maximum Likelihood Estimate of Vanishing Point [28].

An initial solution for $\boldsymbol{v}$ is obtained from the null vector of the matrix $[\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_n]$ via singular value decomposition [28].

An implementation of this algorithm can be found in the *DARPA Image Understanding Environment* software package (©Amerinex Applied Imaging)[1].

[1]For more information and to download the software package go to: http://www.aai.com/AAI/IUE/ (accessed November 2000).

Appendix C

The Levenberg-Marquardt Algorithm

All minimisations in this thesis have been done using the *Levenberg-Marquardt* algorithm [35, 40]. A short description of the Levenberg-Marquardt algorithm is presented by Hartley [17] and Pollefeys [38], and is outlined here. The algorithm is based on the *Newton iteration* method, which is described first.

C.1 Newton Iteration

For a vector relation $\boldsymbol{y} = f(\boldsymbol{x})$, where $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors in different dimensions, with $\hat{\boldsymbol{y}}$ a measured value for $\boldsymbol{y}$, a desired vector $\hat{\boldsymbol{x}}$ needs to be calculated which satisfies the above relation. Stated differently, a vector $\hat{\boldsymbol{x}}$ needs to be calculated which satisfies $\hat{\boldsymbol{y}} = f(\hat{\boldsymbol{x}}) + \hat{\boldsymbol{\epsilon}}$, for which $\|\hat{\boldsymbol{\epsilon}}\|$ is minimal. Newton's iteration method starts with an initial value $\boldsymbol{x}_0$ and refines this value under the assumption that the function f is locally linear [17, 38].

For $\hat{\boldsymbol{y}} = f(\boldsymbol{x}_0) + \boldsymbol{\epsilon}_0$, the function f is approximated at $\boldsymbol{x}_0$ by $f(\boldsymbol{x}_0 + \boldsymbol{\delta}) = f(\boldsymbol{x}_0) + \boldsymbol{J}\boldsymbol{\delta}$, where $\boldsymbol{J} = \frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}}$ is the linear mapping represented by the Jacobian matrix. Then setting $\boldsymbol{x}_1 = \boldsymbol{x}_0 + \boldsymbol{\delta}$, the following relation is obtained:

$$\hat{\boldsymbol{y}} - f(\boldsymbol{x}_1) = \hat{\boldsymbol{y}} - f(\boldsymbol{x}_0) - \boldsymbol{J}\boldsymbol{\delta} = \boldsymbol{\epsilon}_0 - \boldsymbol{J}\boldsymbol{\delta},$$

where $\|\boldsymbol{\epsilon}_0 - \boldsymbol{J}\boldsymbol{\delta}\|$ needs to be minimised. This minimisation can be solved for $\boldsymbol{\delta}$ using linear least-squares:

$$\boldsymbol{J}^T\boldsymbol{J}\boldsymbol{\delta} = \boldsymbol{J}^T\boldsymbol{\epsilon}_0.$$

The above equation is called the *normal equation* [17, 38].

To summarise the above procedure, the solution can be found by starting with an initial estimate $\boldsymbol{x}_0$ and approximating the equation

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \boldsymbol{\delta}_i,$$

where $\boldsymbol{\delta}_i$ is the solution to the normal equations $\boldsymbol{J}^T \boldsymbol{J} \boldsymbol{\delta}_i = \boldsymbol{J}^T \boldsymbol{\epsilon}_i$. Matrix $\boldsymbol{J}$ is the Jacobian at $\boldsymbol{x}_i$ and $\boldsymbol{\epsilon}_i = \hat{\boldsymbol{y}} - f(\boldsymbol{x}_i)$. As pointed out by Hartley [17] and Pollefeys [38], it is possible that this algorithm will converge to a local minimum value or not converge at all, as it depends a lot on the initial estimate $\boldsymbol{x}_0$.

C.2 Levenberg-Marquardt Iteration

This method varies slightly from the Newton method. The normal equations $\boldsymbol{N}\boldsymbol{\delta} = \boldsymbol{J}^T \boldsymbol{J} \boldsymbol{\delta} = \boldsymbol{J}^T \boldsymbol{\epsilon}_0$ are augmented to $\boldsymbol{N}'\boldsymbol{\delta} = \boldsymbol{J}^T \boldsymbol{\epsilon}_0$, where $\boldsymbol{N}'_{ii} = (1+\lambda)\boldsymbol{N}_{ii}$ and $\boldsymbol{N}'_{ij} = \boldsymbol{N}_{ij} \quad \text{for} \quad i \neq j$. The value of λ is initialised to a very small value, usually $\lambda = 10^{-3}$. If the value obtained for $\boldsymbol{\delta}$ by the augmented normal equations reduces the error, then the increment is accepted and λ is divided by 10 before the next iteration. If the error increases, λ is multiplied by 10 and the augmented normal equations are solved until an increment is obtained which reduces the error.

The implementation of the Levenberg-Marquardt algorithm used in this thesis is from the *MINPACK*[1] library and *DARPA Image Understanding Environment* software package (©Amerinex Applied Imaging)[2].

[1]For more information and to download the MINPACK software package go to: http://www.netlib.org (accessed November 2000).

[2]For more information and to download the Image Understanding Environment software package go to: http://www.aai.com/AAI/IUE/ (accessed November 2000).

Appendix D

Triangulation

The process of triangulation is needed to find the intersection of two known rays in space. Due to measurement noise in images and some inaccuracies in the calibration matrices, these two rays will not generally meet in a unique point. This section outlines a linear and a nonlinear method to accurately estimate the best possible point of intersection.

D.1 Linear Triangulation

Let the 3D coordinate in space be $\tilde{\boldsymbol{M}} = [X, Y, Z, W]^T$ and its corresponding image coordinates be $\tilde{\boldsymbol{m}}_{1,2} = [u_{1,2}, v_{1,2}, 1]^T$. Then making use of the pinhole model equation (3.1) and the camera projection matrices relating to the two images, the following two equations can be defined:

$$s_1[u_1, v_1, 1]^T = \boldsymbol{P}_1[X, Y, Z, W]^T, \tag{D.1}$$

$$s_2[u_2, v_2, 1]^T = \boldsymbol{P}_2[X, Y, Z, W]^T, \tag{D.2}$$

where s_1 and s_2 are two arbitrary scalars. Writing the i^{th} row of $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ as $\boldsymbol{p}_{1i}^T$ and $\boldsymbol{p}_{2i}^T$ respectively, the two scalars s_1 and s_2 can be eliminated by knowing the following: $s_1 = \boldsymbol{p}_{13}^T\tilde{\boldsymbol{M}}$ and $s_2 = \boldsymbol{p}_{23}^T\tilde{\boldsymbol{M}}$. The above two equations can then be rewritten in the form:

$$\boldsymbol{A}\tilde{\boldsymbol{M}} = 0, \tag{D.3}$$

where $\boldsymbol{A}$ is a 4×4 matrix:

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{p}_{11}^T - u_1 \boldsymbol{p}_{13}^T \\ \boldsymbol{p}_{12}^T - v_1 \boldsymbol{p}_{13}^T \\ \boldsymbol{p}_{21}^T - u_2 \boldsymbol{p}_{23}^T \\ \boldsymbol{p}_{22}^T - v_2 \boldsymbol{p}_{23}^T \end{bmatrix}.$$

The solution is then the eigenvector of the matrix $\boldsymbol{A}^T \boldsymbol{A}$ associated with the smallest eigenvalue. It is also possible to assume that no point is at infinity, and then $W = 1$. The set of homogeneous equations (D.3) is then reduced to a set of four non-homogeneous equations in three unknowns [52].

D.2 Nonlinear Triangulation

The above linear method can be used as an initial estimate for a nonlinear optimisation problem. Zhang [52] minimises the error measured in the image plane between the observation and the projection of the reconstructed point:

$$\left(u_1 - \frac{\boldsymbol{p}_{11}^T \tilde{\boldsymbol{M}}}{\boldsymbol{p}_{13}^T \tilde{\boldsymbol{M}}}\right)^2 + \left(v_1 - \frac{\boldsymbol{p}_{12}^T \tilde{\boldsymbol{M}}}{\boldsymbol{p}_{13}^T \tilde{\boldsymbol{M}}}\right)^2 + \left(u_2 - \frac{\boldsymbol{p}_{21}^T \tilde{\boldsymbol{M}}}{\boldsymbol{p}_{23}^T \tilde{\boldsymbol{M}}}\right)^2 + \left(v_2 - \frac{\boldsymbol{p}_{22}^T \tilde{\boldsymbol{M}}}{\boldsymbol{p}_{23}^T \tilde{\boldsymbol{M}}}\right)^2. \tag{D.4}$$

The nonlinear minimisation is done using the *Levenberg-Marquardt* algorithm [35, 40], described in appendix C.

Hartley and Sturm [21] define a different minimisation problem. Their method seeks a pair of points that minimises the sum of squared distances subject to the epipolar constraint. But as Zhang [52] points out in his paper, both the linear and nonlinear triangulation methods outlined above will give optimum results.

Bibliography

[1] M.N. Armstrong. *Self-Calibration from Image Sequences*. PhD thesis, Department of Engineering Science, University of Oxford, 1996.

[2] J.R. Beveridge, C. Graves, and C. Lesher. Some Lessons Learned from Coding the Burns Line Extraction Algorithm in the DARPA Image Understanding Environment. Technical Report CS-96-125, Computer Science Department, Colorado State University, October 1996.

[3] S. Birchfield. An Introduction to Projective Geometry (for Computer Vision). Stanford University, March 1998.

[4] J.B. Burns, A.R. Hanson, and E.M. Riseman. Extracting Straight Lines. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(4):425–455, July 1986.

[5] B. Caprile and V. Torre. Using Vanishing Points for Camera Calibration. *International Journal of Computer Vision*, 4:127–140, 1990.

[6] Intel Corporation. Open Source Computer Vision Library Manual. See official website: http://www.intel.com/research/mrl/research/cvlib/, 2000.

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

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

[9] O. Faugeras. Stratification of 3-D vision: projective, affine, and metric representations. *Journal of the Optical Society of America*, 12(3):465–484, March 1995.

[10] O. Faugeras, Q.-T. Luong, and S. Maybank. Camera Self-Calibration: Theory and Experiments. In *Computer-Vision-ECCV'92, Lecture Notes in Computer Science*, volume 588, pages 321–334. Springer-Verlag, 1992.

[11] O. Faugeras, L. Robert, S. Laveau, G. Csurka, C. Zeller, C. Gauclin, and I. Zoghlami. 3-D Reconstruction of Urban Scenes from Image Sequences. *Computer Vision and Image Understanding*, 69(3):292–309, March 1998. Also Research Report No.2572, INRIA Sophia-Antipolis.

[12] A. Fusiello, V. Roberto, and E. Trucco. Efficient stereo with multiple windowing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 858–863, June 1997.

[13] A. Fusiello, E. Trucco, and A. Verri. Rectification with unconstrained stereo geometry. In A.F. Clark, editor, *Proceedings of the British Machine Vision Conference*, pages 400–409. BMVA Press, September 1997. Also Research Memorandum RM/98/12, 1998, Department of Computing and Electrical Engineering, Heriot-Watt University, Edinburgh, UK.

[14] A. Fusiello, E. Trucco, and A. Verri. A compact algorithm for rectification of stereo pairs. *Machine Vision and Applications*, 12(1):16–22, 2000.

[15] G.H. Golup and C.F. van Loan. *Matrix Computations*. John Hopkins University Press, 2^{nd} edition, 1996.

[16] C. Harris and M. Stephens. A Combined Corner and Edge Detector. *Proceedings Alvey Vision Conference*, pages 189–192, 1988.

[17] R. Hartley. Euclidean Reconstruction from Uncalibrated Views. In J.L. Mundy, A. Zisserman, and D. Forsyth, editors, *Applications of Invariance in Computer Vision*, volume 825 of *Lecture Notes in Computer Science*, pages 237–256. Springer-Verlag, 1994.

[18] R. Hartley. In defense of the 8-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(6):580–593, June 1997.

[19] R. Hartley. Kruppa's Equations Derived from the Fundamental Matrix. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):133–135, February 1997.

[20] R. Hartley and J.L. Mundy. The relationship between photogrammetry and computer vision. In E.B. Barrett and D.M. McKeown, editors, *SPIE Proceedings*, volume 1944 of *Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision*, pages 92–105. SPIE Press, September 1993.

[21] R. Hartley and P. Sturm. Triangulation. *Computer Vision and Image Understanding*, 68(2):146–157, 1997.

[22] J. Heikkilä and O. Silvén. A Four-step Camera Calibration Procedure with Implicit Image Correction. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'97), San Juan, Puerto Rico*, pages 1106–1112, 1997.

[23] L. Kitchen and A. Rosenfeld. Gray-level corner detection. *Pattern Recognition Letters*, 1(2):95–102, December 1982.

[24] R. Koch. Automatic Reconstruction of Buildings from Stereoscopic Image Sequences. *Proceedings of Eurographics '93*, 12(3):339–350, September 1993.

[25] R. Koch. 3-D Surface Reconstruction from Stereoscopic Image Sequences. In *Proc. 5th International Conference on Computer Vision, Cambridge, MA., USA*, pages 109–114, June 1995.

[26] S. Laveau and O. Faugeras. Oriented Projective Geometry for Computer Vision. In B. Buxton and R. Cipolla, editors, *Computer-Vision-ECCV'96, Lecture Notes in Computer Science*, volume 1064, pages 147–156. Springer-Verlag, 1996.

[27] D. Liebowitz, A. Criminisi, and A. Zisserman. Creating Architectural Models from Images. In *Proc. EuroGraphics*, volume 18, pages 39–50, September 1999.

[28] D. Liebowitz and A. Zisserman. Metric Rectification for Perspective Images of Planes. In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, pages 482–488, 1998.

[29] D. Liebowitz and A. Zisserman. Combining Scene and Auto-calibration Constraints. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 293–300, September 1999.

[30] M.I.A. Lourakis and R. Deriche. Camera Self-Calibration Using the Singular Value Decomposition of the Fundamental Matrix: From Point Correspondences to 3D Measurements. Technical Report 3748, INRIA Sophia Antipolis, Project Robotvis, 1999.

[31] Q.-T. Luong and O. Faugeras. The Fundamental matrix: theory, algorithms, and stability analysis. *The International Journal of Computer Vision*, 1(17):43–76, 1996.

[32] S.J. Maybank and O. Faugeras. A Theory of Self-Calibration of a Moving Camera. *International Journal of Computer Vision*, 8(2):123–151, August 1992.

[33] R. Mohr and B. Triggs. Projective Geometry for Image Analysis. In *International Symposium of Photogrammetry and Remote Sensing*, Vienna, July 1996.

[34] H. Moravec. Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover. Technical Report CMU-RI-TR-3, Robotics Institute, Carnegie-Mellon University, September 1980.

[35] J.J. Moré. The Levenberg-Marquardt Algorithm: Implementation and Theory. In G.A. Watson, editor, *Numerical Analysis*, volume 630 of *Lecture Notes in Mathematics*. Springer-Verlag, 1977.

[36] M. Pilu. Uncalibrated Stereo Correspondence by Singular Value Decomposition. Technical Report HPL-97-96, Digital Media Department, HP Laboratories Bristol, August 1997.

[37] M. Pollefeys. *Self-Calibration and Metric 3D Reconstruction from Uncalibrated Image Sequences*. PhD thesis, ESAT-PSI, K.U. Leuven, 1999.

[38] M. Pollefeys. Tutorial on 3D Modelling from Images. Tutorial organised in conjunction with ECCV 2000, Dublin, Ireland, June 2000.

[39] M. Pollefeys, R. Koch, and L. van Gool. A simple and efficient rectification method for general motion. In *Proc. 7th International Conference on Computer Vision, Kerkyra, Greece*, pages 496–501, September 1999.

[40] W. Press, B. Flannery, S. Teukolsky, and W. Vetterling. *Numerical Recipes in C*. Oxford University Press, 1979.

[41] P.J. Rousseeuw and A.M. Leroy. *Robust Regression and Outlier Detection*. John Wiley & Sons, New York, 1987.

[42] S. Roy, J. Meunier, and I. Cox. Cylindrical Rectification to Minimize Epipolar Distortion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 393–399, 1997.

[43] G. Scott and H. Longuet-Higgins. An algorithm for associating the features of two images. *Proc. Royal Society London*, B244:21–26, 1991.

[44] J.G. Semple and G.T Kneebone. *Algebraic Projective Geometry*. Oxford University Press, 1979.

[45] S.M. Smith. Reviews of Optic Flow, Motion Segmentation, Edge finding and Corner Finding. Technical Report TR97SMS1, Oxford Centre for Functional Magnetic Resonance Imaging of the Brain (FMRIB), Department of Clinical Neurology, Oxford University, Oxford, UK, 1997.

[46] C.J. Taylor and D.J. Kriegman. Structure and Motion from Line Segments in Multiple Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(11):1021–1032, November 1995.

[47] P. Torr. *Motion Segmentation and Outlier Detection*. PhD thesis, Department of Engineering Science, University of Oxford, 1995.

[48] S. Ullman. *The Interpretation of Visual Motion*. MIT Press, Cambridge, MA, 1979.

[49] C. Zeller and O. Faugeras. Camera Self-Calibration from Video Sequences: the Kruppa Equations Revisited. Technical Report 2793, INRIA Sophia Antipolis, Project Robotvis, 1996.

[50] Z. Zhang. A New Multistage Approach to Motion and Structure Estimation: From Essential Parameters to Euclidean Motion via Fundamental Matrix. Technical Report 2910, INRIA Sophia-Antipolis, France, June 1996.

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

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

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

[54] Z. Zhang, R. Deriche, O. Faugeras, and Q.-T. Luong. A Robust Technique for Matching Two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry. *Artificial Intelligence Journal*, 78:87–119, 1995. Also Research Report No.2273, INRIA Sophia-Antipolis.