

# Stereo Camera Calibration

BRIAN O'KENNEDY

Thesis submitted in partial fulfilment of the requirements for the MSc(Eng), Electronic Engineering degree at the University of Stellenbosch



Supervisor: Prof. B.M. Herbst  
Co-supervisor: Prof. J.G. Lourens

December, 2002

# Declaration

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

# Abstract

We present all the components needed for a fully-fledged stereo vision system, ranging from object detection through camera calibration to depth perception. We propose an efficient, automatic and practical method to calibrate cameras for use in 3D machine vision metrology. We develop an automated stereo calibration system that only requires a series of views of a manufactured calibration object in unknown positions. The system is tested against real and synthetic data, and we investigate the robustness of the proposed method compared to standard calibration practice.

All the aspects of 3D stereo reconstruction is dealt with and we present the necessary algorithms to perform epipolar rectification on images as well as solving the correspondence and triangulation problems.

It was found that the system performs well even in the presence of noise, and calibration is easy and requires no specialist knowledge.

# Opsomming

Ons beskryf al die komponente van 'n omvattende stereo visie sisteem. Die kern van die sisteem is 'n effektiewe, ge-automatiseerde en praktiese metode om kameras te kalibreer vir gebruik in 3D rekenaarvisie.

Ons ontwikkel 'n outomatiese, stereo kamerakalibrasie sisteem wat slegs 'n reeks beelde van 'n kalibrasie voorwerp in onbekende posisies vereis. Die sisteem word getoets met reële en sintetiese data, en ons vergelyk die robuustheid van die metode met die standaard algoritmes.

Al die aspekte van die 3D stereo rekonstruksie word behandel en ons beskryf die nodige algoritmes om epipolêre rektifikasie op beelde te doen sowel as metodes om die korrespondensie- en diepte probleme op te los.

Ons wys dat die sisteem goeie resultate lewer in die aanwesigheid van ruis en dat kamerakalibrasie outomaties kan geskied sonder dat enige spesialis kennis benodig word.

# Acknowledgements

- The Lord, for giving me the abilities
- My Parents for bringing me this far in life.
- Prof. Ben Herbst for his support and guidance.
- Stone Three Signal Processing Pty, and especially Dave Weber for support; intellectually, motivationally and financially
- All the people special to me, they know who they are.

# Contents

<b>Abstract</b>	<b>i</b>
<b>Declaration</b>	<b>ii</b>
<b>List of figures</b>	<b>viii</b>
<b>List of tables</b>	<b>ix</b>
<b>1 Introduction</b>	<b>1</b>
1.1 General Overview . . . . .	1
1.2 Camera Calibration . . . . .	2
1.3 The 3D reconstruction problem . . . . .	2
1.4 Outline of Thesis . . . . .	3
<b>2 Calibration Object Location</b>	<b>5</b>
2.1 Introduction . . . . .	5
2.2 Image Acquisition . . . . .	6
2.3 Requirements . . . . .	7
2.4 Algorithm . . . . .	7
2.4.1 Corner Detection . . . . .	12
2.4.2 Sub-Pixel Accuracy . . . . .	14
2.5 Results . . . . .	15
2.5.1 Noncoplanar . . . . .	15
2.5.2 Coplanar . . . . .	16
<b>3 Camera Calibration</b>	<b>17</b>
3.1 Introduction . . . . .	17
3.2 Literature Overview . . . . .	17
3.2.1 Classical Calibration Methods . . . . .	18
3.2.2 Self Calibration . . . . .	19
3.3 Camera Model . . . . .	20
3.3.1 Intrinsic Parameters . . . . .	20

3.3.2	Extrinsic Parameters . . . . .	23
3.3.3	Full Mathematical Model . . . . .	23
3.4	Intrinsic Calibration Algorithm . . . . .	25
3.4.1	Nonlinear Optimization . . . . .	28
3.5	Results . . . . .	30
3.5.1	Radial Distortion . . . . .	31
3.5.2	Robustness against noise . . . . .	33
<b>4</b>	<b>Stereo Camera Calibration</b>	<b>35</b>
4.1	Introduction . . . . .	35
4.2	Overview of our calibration method . . . . .	36
4.3	Epipolar Geometry . . . . .	37
4.4	Essential Matrix . . . . .	38
4.5	Fundamental Matrix . . . . .	39
4.5.1	Normalization . . . . .	40
4.5.2	Parametrization . . . . .	41
4.5.3	Non Linear Criteria . . . . .	42
4.6	Epipolar Lines . . . . .	42
4.7	Factorization . . . . .	44
4.8	Scale Factor . . . . .	45
4.9	Results . . . . .	45
<b>5</b>	<b>Triangulation</b>	<b>47</b>
5.1	Introduction . . . . .	47
5.2	Mid-Point Method . . . . .	48
5.3	Results . . . . .	49
<b>6</b>	<b>Disparity</b>	<b>51</b>
6.1	Introduction . . . . .	51
6.2	Correspondence problem . . . . .	52
6.2.1	Feature based correspondence . . . . .	53
6.2.2	Correlation based Technique . . . . .	53
6.3	Constraints . . . . .	54
6.3.1	Epipolar Constraint . . . . .	55
6.3.2	Left-Right Constraint . . . . .	62
6.3.3	Continuity . . . . .	63
6.3.4	Validity Measure . . . . .	63
6.4	Speed . . . . .	63
6.5	Results . . . . .	64

<b>7 Applications</b>	<b>66</b>
7.1 Computer Vision Metrology . . . . .	66
7.2 3D Tracking . . . . .	68
<b>8 Conclusion</b>	<b>70</b>

# List of Figures

1.1	2D Image of a scene . . . . .	2
2.1	A Noncoplanar Calibration Object . . . . .	5
2.2	Algorithm Outline . . . . .	8
2.3	Detection: Captured Image . . . . .	8
2.4	Detection: Binarized . . . . .	9
2.5	Detection: Noise Outlines . . . . .	9
2.6	Hough Transform - Well defined . . . . .	10
2.7	Hough Transform - Badly defined . . . . .	11
2.8	Detection: Corner Locations . . . . .	12
2.9	Detection: Horizontal and Vertical Gradients . . . . .	13
2.10	Detection: Sub-Pixel Corners . . . . .	15
2.11	Noncoplanar calibration sequence . . . . .	16
2.12	Coplanar calibration sequence . . . . .	16
3.1	Pinhole Camera model . . . . .	20
3.2	3D Coordinate to Image transformation . . . . .	24
3.3	CMU image with ground truth . . . . .	30
3.4	Radial Distortion . . . . .	32
3.5	Radially distorted and undistorted images . . . . .	32
3.6	Comparison of Single camera calibration methods with Noisy data . . . . .	33
4.1	Stereo Calibration algorithm . . . . .	37
4.2	Epipolar Geometry . . . . .	38
4.3	Epipolar Lines . . . . .	43
4.4	Epipolar Lines . . . . .	44
4.5	Translation Vector errors . . . . .	46
4.6	Rotation Angle errors . . . . .	46
5.1	Triangulation with non-intersecting rays . . . . .	48
5.2	3D position errors . . . . .	49
5.3	2D projection errors . . . . .	50

6.1	Correspondence problem . . . . .	51
6.2	Planar Epipolar Rectification . . . . .	56
6.3	Unrectified image planes . . . . .	58
6.4	Rectified image planes . . . . .	59
6.5	Epipolar Unrectified images . . . . .	61
6.6	Epipolar Rectified Images . . . . .	61
6.7	Left-Right Constraint . . . . .	62
6.8	Validity Measure . . . . .	63
6.9	Synthetic Cube for Disparity Calculation . . . . .	64
6.10	Disparity and Validity for Synthetic Cube . . . . .	64
6.11	Aerial Pentagon view for Disparity Calculation . . . . .	65
6.12	Disparity and Validity for Pentagon . . . . .	65
7.1	Stereo Images for Measurements . . . . .	66
7.2	Index of Measurement Points . . . . .	67
7.3	Selected images of left camera person tracking . . . . .	68
7.4	Top View of track . . . . .	69

# List of Tables

3.1	Comparison of intrinsic parameters . . . . .	31
4.1	Fundamental Matrix results . . . . .	43
7.1	Feature Coordinates . . . . .	67
7.2	Comparison of Physical Distances and Triangulation . . . . .	68

# Chapter 1

## Introduction

The objective of this thesis is to present an automatic camera calibration software library which relies on the minimum of human intervention. It also deals with the 3D reconstruction problem from stereo images, notably triangulation and the recovery of dense disparity maps.

Researchers have been investigating methods of obtaining 3D models from 2D images for many years. The emphasis has however shifted from robotic guidance and inspection systems to cover a much broader field of applications. The technical expertise and operating conditions of these new applications can often not be matched with the requirements of existing systems. This creates a demand for an easy to use and robust system of calibrating cameras as well as generating 3D reconstructions, which is the problem this thesis addresses.

### 1.1 General Overview

The field of 3D reconstructions of scenes has been actively researched, and many different solutions have been proposed. These can be roughly divided into active and passive systems. Active systems control the lighting in a scene in an effort to simplify the problem, but this also restricts the applications. This can for example be in the form of projected laser dots which can be easily detected in the image. Passive systems are completely non-intrusive but present a plethora of new problems and can be very computationally expensive. Passive systems can also be very dependent on the structure and appearance of the scene.

Some examples of active techniques are the grid projection method proposed by Proesmans et al [31] and the shadow based approach by Bouquet and Perona [9] which is able to extract textured 3D shapes.

Many passive techniques exist and the field is constantly expanding. The main differences between the methods are the required level of calibration and the amount of



Figure 1.1: *2D Image of a scene*

interaction that is required.

## 1.2 Camera Calibration

Camera calibration in the context of 3D machine vision is the process of determining the internal geometric and optical characteristics of a camera, which is known as the intrinsic parameters. It can also involve finding the 3D position and orientation of the camera frame relative to some world coordinate system, and this is known as the extrinsic parameters.

With an uncalibrated camera it is not possible to derive accurate knowledge about the physical dimensions or locations of observed objects. Some applications demand this and that is the main reason for camera calibration. The scene observed by the camera is related via a mathematical model to the image obtained. This model is what we aim to solve with camera calibration. This field has been intensively studied for many years by photogrammetrists and once the computing power became available, by computer scientists. A wide range of calibration schemes have been proposed, and will be discussed in more detail in chapter 3.

## 1.3 The 3D reconstruction problem

The 3D reconstruction problem entails obtaining a three dimensional model of an observed scene. In this thesis we consider reconstruction from stereo images. An image like in figure 1.1 tells us a lot about the observed scene. Humans are able to derive 3D shapes due to our *a priori* knowledge of the structure, and using such cues as shading. There is not enough information to make an accurate 3D reconstruction of the scene without making a number of assumptions. This is due to the image formation process which projects a

three dimensional scene onto a two dimensional plane. This implies that a specific feature point on the image must lie on the associated line of sight, but does not define where along this line of sight the point lies. If two images from different points of view are available it is possible to intersect these two lines of sight and triangulate the point in space. This is the essence of stereo vision, which is the main thrust of this thesis.

This process is called triangulation and has a number of requirements. These are

- Camera model parameters relating scene to image (intrinsic parameters)
- Relative pose of the cameras for the two views (extrinsic parameters)
- Corresponding features on left and right images

Note that different viewpoints are not the only method of depth extraction from images. It is also possible to use shading, texture, silhouette and focus to give some hint about depth [29].

## 1.4 Outline of Thesis

Chapter 2 shows how we implemented an automatic feature extraction algorithm for our calibration objects. These calibration objects are needed by the calibration routines for intrinsic and extrinsic camera calibration, and are objects with known physical dimensions and features. The algorithm detailed here automatically locates the object in the scene, extracts the corner positions to sub-pixel accuracy and presents them to the calibration routines in the right order.

Chapter 3 describes a well known calibration method proposed by Roger Tsai [35] which we use to determine the intrinsic parameters of our cameras. This method was chosen due to its proven accuracy and effectiveness in industry. This step is very important for our stereo calibration routine since this is where we determine and correct the radial distortion effects of the camera lens which distorts the epipolar plane to an epipolar curve.

Chapter 4 details our stereo calibration method to determine the extrinsic parameters. This method is based on the epipolar geometry which defines the relation between two cameras. The central part of the algorithm is the fundamental matrix, and we clearly show the derivation and calculation of this matrix. The method we propose utilises multiple images of a calibration object, and combines all the feature points into one large *virtual object*. We bypass the traditional method of pose estimation for each image by using the constraints introduced by the epipolar geometry. This results in a flexible, robust and accurate stereo calibration algorithm.

Chapter 5 explains the concept of triangulation. We detail the method we have chosen to use, namely the *mid-point* method. Our motivation for doing so is explained, and we investigate the robustness of the method.

Chapter 6 deals with the correspondence problem, which entails finding corresponding features between two images. These corresponding features are needed for the triangulation algorithm to determine depth. We use the well known cross correlation method with additional constraints based on the epipolar geometry. This enables us to reduce the two dimensional search to an one dimensional search through epipolar rectification, and thus obtain dense disparity maps.

Chapter 7 shows some possible applications of the system we have developed, as well as some experimental results.

Our conclusions are drawn in chapter 8.

## Chapter 2

# Calibration Object Location

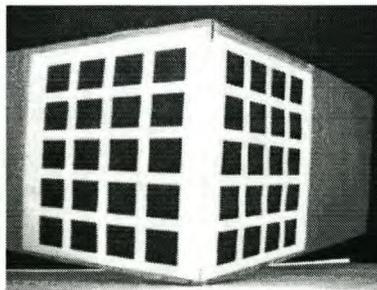
### 2.1 Introduction

The camera calibration literature can be roughly divided into two categories. Firstly there are those that use some form of calibration object with known physical dimensions. The second are those which employ the features naturally found in scenes to perform what is termed *self-calibration*. Examples of the latter can be found in [25] [40] [26] and are mostly based on Kruppa's equations. The disadvantage of these methods is that the calibration can only be done up to an unknown scale factor, which implies that the resultant model can not be used for any application where exact physical measurements are required.

The classical calibration method makes use of a calibration object with known physical dimensions as in figure 2.1. This supplies a frame of reference from which to determine the scale factor, as well as greatly simplifying the calibration routine.

A great number of calibration objects have been used by various researchers. The main difference is whether the calibration routine requires coplanar or noncoplanar data, which requires that the feature points, respectively, must or must not all lie on the same plane in space. This is a direct result of the mathematical approach implemented in the calibration.

Taking the above factors into account, the calibration object is designed purely for ease



**Figure 2.1:** *A Noncoplanar Calibration Object*

of extraction by image processing routines. Objects range from flat planes with distinctive features to 3D structures with reflective markers. The disadvantage of reflective markers is that they require a specific light source, which changes the acquisition process from passive to active. They do however make the extraction process much simpler.

Our calibration object was chosen as shown in figure 2.1. This is a noncoplanar object, with 20 black squares per plane which gives a total of 160 detectable corners. This is the object we use for the single camera calibration to determine intrinsic parameters. For the stereo calibration we used a coplanar object which is more easily visible by both cameras at the same instant.

## 2.2 Image Acquisition

The first step to calibration is acquiring the images of the calibration object. Our system consisted of a Linux [5] PC with two Zoran [7] framegrabbers using the BT848 chip set connected to two Jai-1021 [3] black and white CCD cameras. We assume a fixed focal length throughout the calibration process.

For the single camera calibration we used the noncoplanar object shown in figure 2.1. We capture a sequence of frames with somebody holding the object and moving in front of the camera until it has a clear view of the object. This sequence of images is then fed into the calibration object location algorithm. It attempts to locate the object in each image, and if successful returns the exact sub-pixel coordinates of the features; the corners of the black squares from our calibration object.

For the stereo camera calibration the process is very similar, but now we capture from the left and right cameras simultaneously, using a coplanar object. The object is simply moved throughout the field of view of both cameras, the only requirement being that it must be visible to both at the same time instant. It must be clearly stated that our stereo calibration routine does not require any specific object, merely an easily trackable object with a single known dimension.

The simultaneous capture of left and right cameras is extremely important to the stereo calibration routine. It makes the assumption that both images are captured at the same instant in time. Experiments have shown that even a difference of 100 ms between the left and right camera captures can cause instabilities in the calibration, since the object is moving and the cameras observe it at different physical positions, resulting in an unsolvable system. We achieved (near) simultaneous capture by sending the BT848 chip sets the capture trigger as close to each other as possible. This resulted in capture times within 5 ms of each other, which proved to be sufficient. The cameras both sync to the same 50 Hz signal which is supplied by the electricity provider.

## 2.3 Requirements

Camera calibration has mostly been performed in the controlled environments of laboratories or by people with the required knowledge to supply the necessary human input [34]. We have tried to eliminate this and supply a calibration method which can be easily and quickly performed in industry without needing any specialised knowledge of the system. Thus automation of the calibration process was a major requirement. Another major factor is that the accuracy of the camera calibration depends greatly on the exact extractions of the object features. Thus it is very important to ensure the accuracy and validity of this data.

The requirements of this system are as follows:

- Automatic extraction of calibration object features
- Sub Pixel accuracy on corner extraction
- Robustness against cluttered environments and varying lighting conditions
- Low error rate

Our location algorithm is based on a number of standard image processing techniques and will be discussed in the next section. The geometric simplicity of the object enables us to locate it in most views where it is clearly visible. Due to the requirement of low error rates we reject possible solutions if the algorithm detects uncertainties. For instance, we use the noncoplanar object for intrinsic calibration of the cameras, for which the optimum is that the object fills as much of the image as possible. In the case that we detect the object too far away from the camera it is rejected, since inaccuracies in the extraction would result in a sub-optimal camera model.

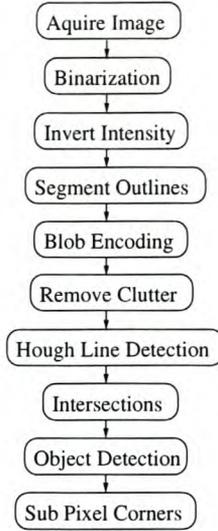
## 2.4 Algorithm

We will present this in an algorithmic fashion. The basic structure of the algorithm is shown in figure 2.2, of which each step shall now be explained.

The only *a priori* information available is the structure of the calibration object. The important characteristics are the dimensions of the squares, as well as their arrangement. We would typically start with an input as in figure 2.3 which has been acquired by the computer.

### **Binarization**

The first step is to binarize the image in order to simplify the location procedure.



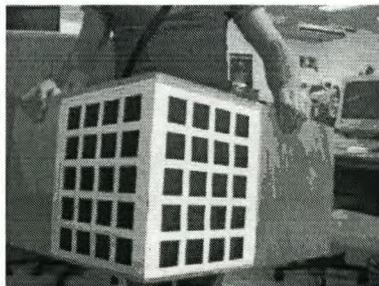
**Figure 2.2:** *Algorithm Outline*

A simple threshold proves to be ineffective in some cases where the background dominates the lighting, resulting in incomplete thresholding of the calibration object. We opted to use a locally adaptive binarization method, *Niblack binarization*, as proposed by Wayne Niblack [28]. This is a thresholding method by which the threshold is dynamically adjusted, and resulted in correct segmentation of the object in all our experiments.

The idea of this binarization method is to vary the threshold over the image, based on the local mean and local standard deviation. The threshold  $T$  at pixel  $(x, y)$  is calculated as:

$$T(x, y) = m(x, y) + ks(x, y) \quad (2.1)$$

where  $m(x, y)$  and  $s(x, y)$  are the sample mean and standard deviation values respectively in a local neighbourhood of  $(x, y)$ . The size of this neighbourhood should be small enough to preserve local details, but also large enough to suppress noise. We used a window size of  $55 \times 55$ . The value of  $k$  is used to adjust how much of the object boundary is taken as a part of the given object. *Trier and Jane* [32] recommended a value of



**Figure 2.3:** *Detection: Captured Image*



Figure 2.4: *Detection: Binarized*

$-0.2$ , which gives well separated objects. The result from this segmentation is shown in figure 2.4.

### Outline Detection

The segmentation is followed by an image inversion and then a binary image processing routine to remove the interior of any solid area. The outline detection routine is a very simple binary morphological filter which removes any pixel which has all its 4-connected neighbours set to one. The 4-connected neighbours are simply the pixels above, below, left and right of the pixel in question. This leaves us with an image like the left of figure 2.5, which clearly denotes the outlines of objects.

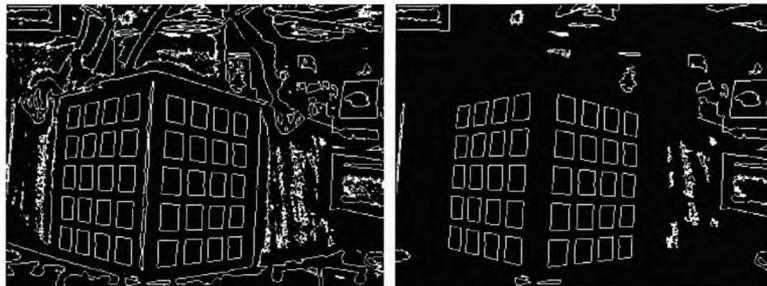


Figure 2.5: *Detection: Noise Outlines*

### Blob Encoding

The outlines detected are then encoded using a blob detection scheme. This entails that every object in the image is assigned an index value. An object is defined as any collection of pixels which are all 8-connected. The *8-connected* neighbours of a pixel are the eight surrounding it. With every object, which in our case are outlines, assigned an index number, we are able to classify them on length and image area.

All outlines which are too small or big are discarded, and the resultant image is shown on the right of figure 2.5. The thresholds for valid size are based on how small a square could be and still provide data which is accurate enough for the calibration routine. The

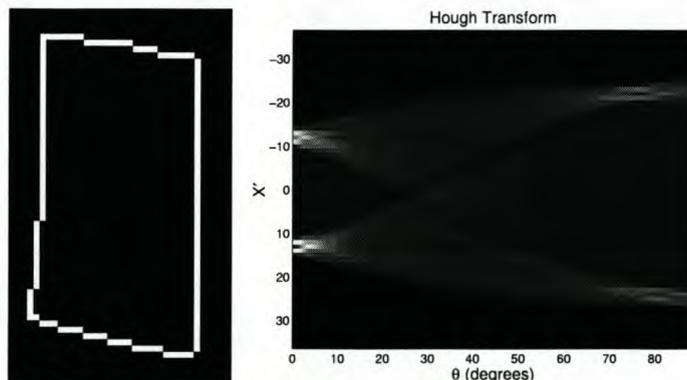
upper limit is determined by the assumption that the entire object needs to fit inside the image combined with the *a priori* knowledge of the object dimensions.

## Hough Transform

The outline data now available needs to be processed to determine which outlines are squares and which are due to the background clutter. Note that quite often the background would also contain some squares, which are classified as squares by this routine but discarded later in the algorithm. The square detection is done by means of a Hough transform. The Hough transform is a very popular algorithm to detect lines and simple curves in images [33]. The key idea is to map a difficult pattern detection problem to a simple peak detection problem.

Any line,  $y = mx + n$ , is identified by an unique parameter pair,  $(m, n)$ . Therefore a line is represented by a point in the  $m, n$  plane (the *parameter space*). Conversely, any point  $\mathbf{p} = [x, y]^T$  in the image corresponds to a line  $n = x(-m) + y$  in parameter space, which, as  $m$  and  $n$  vary, represents all possible lines through  $\mathbf{p}$ . This enables us to divide the  $(m, n)$  plane into a finite number of cells, each of which can be seen as an accumulator. For a given point  $\mathbf{p}$  in the image we increment all accumulators on the corresponding line in the parameter space. Therefore the image lines are simply detected as peaks of  $c(m, n)$  in the parameter space, where  $c(m, n)$  are the accumulator cells.

Both  $m$  and  $n$  can take on values in the range  $[-\infty, \infty]$ , which implies that we cannot sample the whole parameter space. Moreover, the  $(m, n)$  parametrization does not capture the bundle  $x = k$ , with  $k$  a constant. The *polar representation*  $p = x \cos \theta + y \sin \theta$  where  $p$  represents the distance between the image origin and the line, and  $\theta$  the line orientation solves both problems.

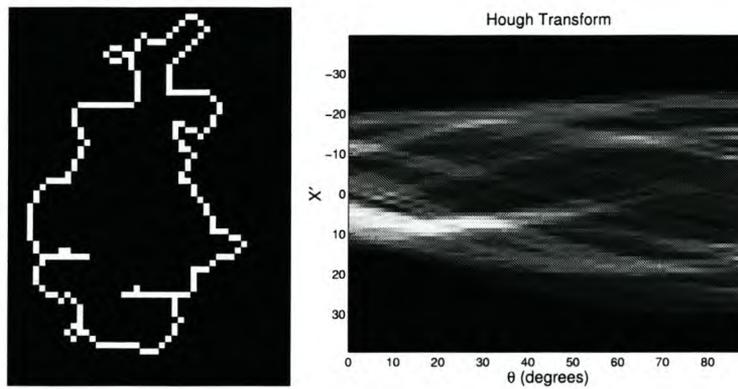


**Figure 2.6:** *Hough Transform - Well defined*

We do the Hough transform on each contour separately, and attempt to extract the four most dominant lines in the contour. This enables us to approximate the contour with a four sided polygon. This was also implemented by a line-simplification algorithm due to

Ramer-Douglas-Peucker [18]. We found that our Hough transform based algorithm with some post-processing gave more consistent results.

We now show two examples of the Hough transform approach. Figure 2.6 shows the outline of a square together with its Hough transform. The horizontal axis defines the angle of detected lines. This angle is measured with vertical being  $0^\circ$  and horizontal  $90^\circ$ . The vertical axis denotes distance from the origin perpendicular to the line. The two clearly visible peaks on the left correspond to the two vertical lines of the outline. The peaks on the right of the transform correspond to the two horizontal sides of the outline. Figure 2.7 shows the Hough transform for a outline which is not a square. It can be clearly seen that this result has no well defined peaks.



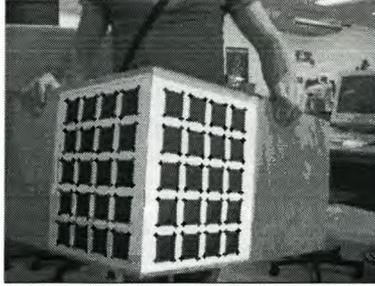
**Figure 2.7:** *Hough Transform - Badly defined*

## Intersections

Using the result from our Hough transform it is easy to pick out the 4 dominant peaks and reconstruct the outline of the square. From the reconstruction we would then be able to see if our original outline is in fact a square. This we do via a distance measure between our reconstructed square and the original outline. The corner positions can now be obtained by calculating the intersection points of these four lines. We use this method to detect all square outlines in the image, and discard all the rest. Note that these square outlines which are kept could possibly include objects in the background which have square features, such as the computer screen in figure 2.8. The next section explains how we disregard this background clutter.

## Object Detection

Using this knowledge of all the square locations in the image, combined with the *a priori* knowledge of the grid dimensions, our algorithm does a brute force search to determine where in the image the calibration grid can be. This is done by searching for the correct number of squares with consistent dimensions forming a planar grid with the known



**Figure 2.8:** *Detection: Corner Locations*

number of rows and columns. Due to the small search space this can be done extremely fast. What we now have is the pixel coordinates of all the corners on our calibration grid as shown in figure 2.8.

One of the requirements however was to locate the corners to sub-pixel precision. This is done via a Harris-Plessey [15] corner detector. More detail on this is supplied in the next section.

### 2.4.1 Corner Detection

Moravec [27] developed the idea of using *points of interest*. They are defined as occurring when there are large intensity variations in every direction. This definition is realised by computing an unnormalized local autocorrelation in four directions and taking the lowest result as the measure of interest. This response is thresholded and local non-maxima are then suppressed. However, some problems with the Moravec operator have been identified: the response is anisotropic and noisy, and the operator is sensitive to strong edges.

Harris and Stephens [15] described what has become known as the Harris-Plessey corner detector. The Harris-Plessey corner detector was developed as a set of enhancements to the Moravec interest operator. The problem of detecting corners can be formulated in terms of the curvature properties of the local image brightness autocorrelation function where the curvature information is represented by the Hessian matrix. The autocorrelation is useful for characterising how the brightness values change in the neighbourhood of a location. At a corner or an isolated brightness peak all shifts will result in a large change in the autocorrelation function at that location. An estimate of the brightness spatial variation function is given by

$$E(x, y) = \sum_{u,v} |I_{x+u,y+v} - I_{u,v}|^2 \quad (2.2)$$

Where  $E$  is the average changes of image intensity produced by a shift  $(x, y)$ , and  $I$  denotes the image intensity.

A first order approximation of  $E$  is given by:

$$E(x, y) = Ax^2 + By^2 + 2Cxy \quad (2.3)$$

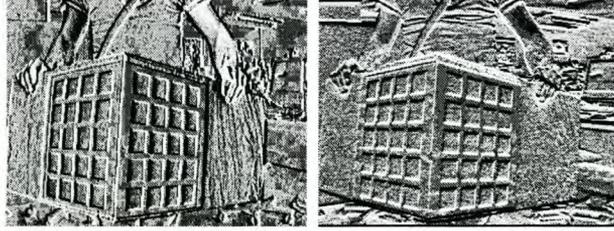
where  $A, B, C$  are approximations of the second order directional derivatives, which are computed by  $X^2, Y^2, XY$  respectively convolving with  $w$

$$A = X^2 \otimes w, \quad B = Y^2 \otimes w, \quad C = XY \otimes w \quad (2.4)$$

where  $\otimes$  is the 2D convolution operator and  $w$  is a Gaussian smoothing circular window to dampen the noise sensitivity of the first order directional image derivatives and defined as:

$$w = \frac{1}{2\pi\delta^2} e^{-\frac{x^2+y^2}{2\delta^2}} \quad (2.5)$$

$X$  and  $Y$  are the first order directional image derivatives, which are approximated by convolving the image with a finite difference operator of the form  $[1, 0, -1]$  in the X direction and  $[1, 0, -1]^T$  in the Y direction. The results are shown in figure 2.9.



**Figure 2.9:** *Detection: Horizontal and Vertical Gradients*

$$\begin{aligned} X &= I \otimes [1, 0, -1] \approx \frac{\partial I}{\partial x} \\ Y &= I \otimes [1, 0, -1]^T \approx \frac{\partial I}{\partial y} \end{aligned} \quad (2.6)$$

Then (2.3) can be rewritten as:

$$E(x, y) = [x, y]M[x, y]^T \quad (2.7)$$

where the matrix  $M$  is an approximation of the Hessian matrix for  $E$ :

$$M|_{(x,y)} = \begin{bmatrix} A & C \\ C & B \end{bmatrix}_{(x,y)} \quad (2.8)$$

$E$  is closely related to the image's local auto correlation function. The principal curvatures of the image brightness auto correlation at a point can be approximated by the

eigenvalues of the approximated 2x2 Hessian matrix  $M$  defined at that point. If both eigenvalues of this matrix are large, implying that this local correlation function is sharply peaked, then a shift in any direction will increase  $E$ . This means that the local grey patch cannot be moved in any direction without a significant change in grey level. This indicates that the window is at a corner.

The determinant of the approximated Hessian matrix,  $\det[M]$ , is proportional to the product of the principal curvatures. The Harris-Plessey corner detector is given by the following operator where a large value of  $R$  signals the presence of a corner. We look for the strongest response of this function within a fixed search area around our estimated position of the corner which is supplied by the location algorithm.

$$R(x, y) = \det[M] - k(\text{trace}[M])^2 \quad (2.9)$$

where

$$\begin{aligned} \text{trace}[M] &= A + B \\ \det[M] &= AB - C^2 \end{aligned} \quad (2.10)$$

and  $M$  is calculated as in (2.8) for this image window. A positive value of the scalar  $k$  is used to ensure that the Gaussian curvature approximate  $\det[M]$  is valid. A value of  $k = 0.04$  is widely specified [40] [15] to provide the best results.

## 2.4.2 Sub-Pixel Accuracy

As Harris's corner detector algorithm only allows us to locate a corner to pixel accuracy, some extra effort needs to be made to recover the corner to sub-pixel accuracy.

The corner detector returns the corner position as an integer value,  $(x, y)$ . However, the actual corner may lie slightly away from this position, and to determine this we interpolate image grey values in the areas near detected corners. For this interpolation we know that the ideal 1D interpolation filter in frequency space is the box, but the inverse transform of this finite-support spectrum is an infinite impulse response *sinc* function. What we prefer is a filter that comes close to the box in frequency space, but still has a finite, reasonable width in signal space. A popular choice is the *Gaussian window*, and we use this as an interpolation filter to reconstruct the gray values in the required areas. The 2D Gaussian interpolation filter is

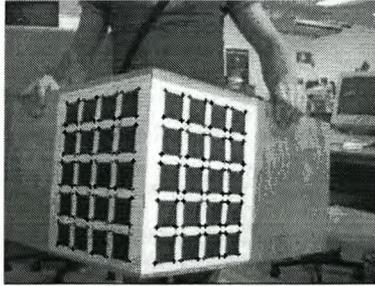
$$g(x, y) = \frac{1}{k_1} e^{-\frac{(x-x_0)^2 + (y-y_0)^2}{2\sigma^2}} \quad (2.11)$$

where  $(x_0, y_0)$  is the centre of the filter. As the Gaussian filter has the property of *separability*, it can be written as  $g(x, y) = g_1(x)g_2(y)$ , where

$$\begin{aligned} g_1(x) &= \frac{1}{\sqrt{k_1}} e^{-\frac{(x-x_0)^2}{2\sigma^2}} \\ g_2(y) &= \frac{1}{\sqrt{k_1}} e^{-\frac{(y-y_0)^2}{2\sigma^2}} \end{aligned} \tag{2.12}$$

This leads to a simple implementation, since it can be implemented as a 1D interpolation filter followed by another perpendicular 1D interpolation filter. After the interpolation the corner detection is used again at these interpolated areas, and corner positions up to 1/10th of a pixel can be detected [22] [15].

The results of our calibration object detection algorithm is a set of estimates of corner positions. These corners are the four corners of each black square, and are normally correct to within 2 pixels. We then use this sub-pixel detector to refine our estimate to an accurate sub-pixel measurement of the corner position as shown in figure 2.10. These are then used as input to our calibration routines.



**Figure 2.10:** *Detection: Sub-Pixel Corners*

## 2.5 Results

### 2.5.1 Noncoplanar

We show a typical series of images of the noncoplanar object here. This sequence would be used to calibrate the intrinsic parameters of a single camera. In this case it is beneficial to have the object as close as possible to the camera to improve the accuracy of the intrinsic parameters. In figure 2.11 we show 9 images, in which we were able to extract the corner positions in the first two rows. The last row shows images where the algorithm was unable to locate the corner positions due to the squares being too small.

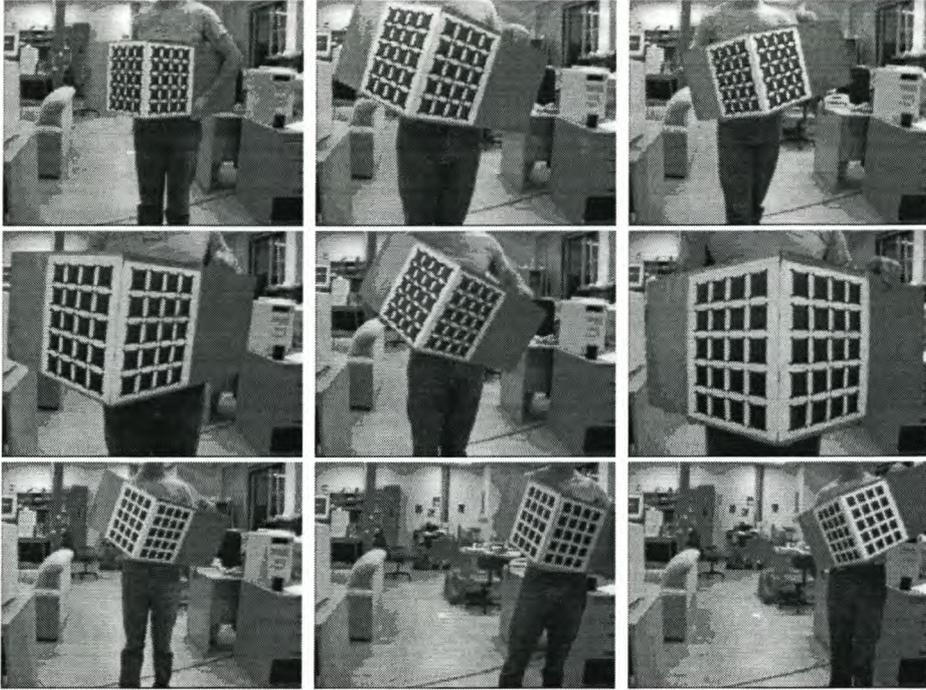


Figure 2.11: *Noncoplanar calibration sequence*

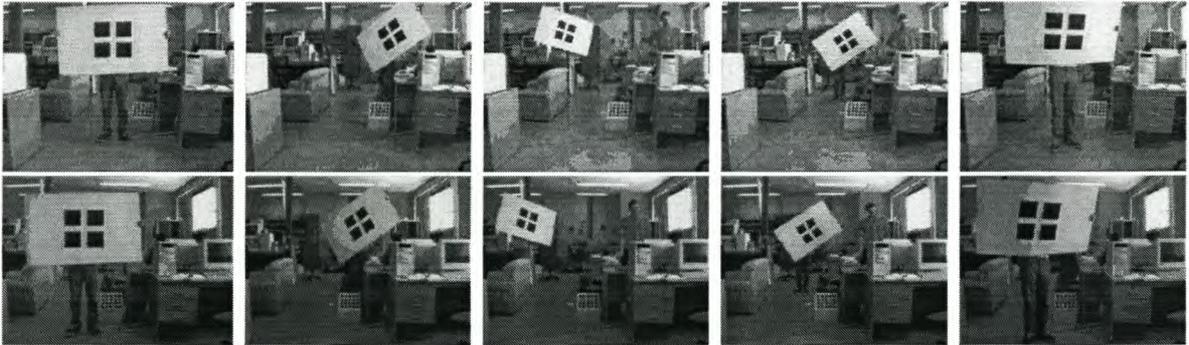


Figure 2.12: *Coplanar calibration sequence*

### 2.5.2 Coplanar

The image sequence showed in figure 2.12 is a typical sequence used to calibrate the extrinsic parameters of a stereo rig. The ideal here is to have a calibration object span the entire field of view of the rig. We show five stereo images where the extraction algorithm could locate the corner positions accurately. The upper and lower rows correspond to the left and right cameras respectively.

We have met the requirements of the system, namely automatic extraction of the calibration object features, robustness and sub-pixel accuracy.

# Chapter 3

## Camera Calibration

### 3.1 Introduction

Camera calibration is an important task in computer vision. The purpose of camera calibration is to establish the relation between 3D world coordinates and their corresponding image coordinates. This enables us to infer 3D information from 2D information and vice versa. Thus camera calibration is a prerequisite for any application where 3D information needs to be related to 2D information. The applications of calibrated cameras are endless, ranging from military to archaeology to medical.

Camera calibration can be divided into two sections, namely intrinsic and extrinsic calibration. Intrinsic calibration solves for the parameters which characterise the inherent properties of the camera and optics, such as focal length and distortion coefficients. Extrinsic calibration is concerned with the translation and rotation of the camera relative to some external coordinate system.

This chapter deals primarily with the calibration of intrinsic parameters of a single camera for which we use a well known calibration method proposed by Roger Tsai [35]. We will give an overview of camera calibration techniques, describe our camera model and calibration method. Finally we will demonstrate the results obtained.

### 3.2 Literature Overview

There are different methods of camera calibration available, depending on factors like the accuracy required, the level of human interaction and whether reconstructions need to be metric. Classical calibration algorithms mostly rely on a calibration object with known dimensions which is placed in the camera field of view. This enables us to fully calibrate the camera, resulting in reconstructions which agree with physical measurements.

Some more recent algorithms rely on features extracted from the scene to calibrate the cameras [25]. These have the advantage that there is no calibration object and no human

intervention is needed, but these models can only reconstruct the scene up to an unknown scale, and relies on the accurate extraction of features in the viewed scene, which are not always available. We divide the calibration literature roughly into the classical methods which require a calibration object, and the newer methods termed *self-calibration*.

### 3.2.1 Classical Calibration Methods

These methods have been studied for a long time by photogrammetrists, and since more computing power has become available have become the domain of computer vision researchers. These methods can be divided into three main groups [34].

#### Approach I: Full scale nonlinear optimization

The transformation between 3D world coordinates and 2D image coordinates is a nonlinear function of all the calibration parameters. Approach I is the classical approach which attempts to do nonlinear optimization to obtain the best estimate of the calibration parameters by minimizing the residual error of the reprojection of known 3D coordinates using the nonlinear equations. The main advantage is that no approximation is involved and that the camera model can be quite elaborate. The problem is the same as that associated with any full scale nonlinear optimization, namely that it is computationally very intensive and requires a good initial guess.

This method is unsuitable to our application due to the requirement of an initial guess, and speed to due the intensive computations.

#### Approach II: Computing perspective transformation matrix

Although the equations characterising the transformation from 3D coordinates to 2D image coordinates are nonlinear functions of the camera parameters, it is possible to linearize the problem by ignoring lens distortion and treating the coefficients of the 3x4 perspective transformation matrix as unknowns. These coefficients are functions of the camera parameters. Given a set of known 3D world coordinates (such as supplied by a calibration object) together with their 2D image coordinates it is possible to solve for the least squares solution of the set of overdetermined linear equations. This enables us to obtain the projection matrix and thus also the camera parameters. The advantage of this method is that no initial estimate is needed and thus automation is possible.

A number of problems exist with this method:

- A number of researchers have found that ignoring lens distortion is unacceptable when doing 3D measurements. [12]
- The resultant rotation matrix is usually not orthonormal.

- The calibration points cannot be coplanar. Allowing calibration points to be coplanar greatly simplifies the construction of calibration objects as well as the feature extraction routines, but due to the linearization in this method coplanar points cause a singularity problem.

The method we use for calibration falls into this category, while overcoming the problems listed here through clever modifications. More detail will be supplied later in this chapter.

### Approach III: Two-plane method

Two-plane methods model the transformation from the image coordinates to the 3D coordinates on several calibration planes to be linear, and once each individual linear transformation is estimated, the rest of the 3D points within these calibration planes are interpolated. The advantages are similar to those of the category II methods i.e., the computation is linear. However, all the problems of category II come into play here too. In addition, the number of unknowns to solve is at least 24 (12 per plane), which is much larger than the degrees of freedom. Some work on this topic can be found in [14] and [19].

## 3.2.2 Self Calibration

In the case of self calibration, a known object in the scene is replaced by an abstract object, the *absolute conic*. The absolute conic, denoted by  $w_\infty$ , is a particular conic in the plane of infinity, which is invariant to transformations of 3D space. This implies that the image of the absolute conic is independent of the position and orientation of the camera, and depends solely on the intrinsic parameters of the camera.

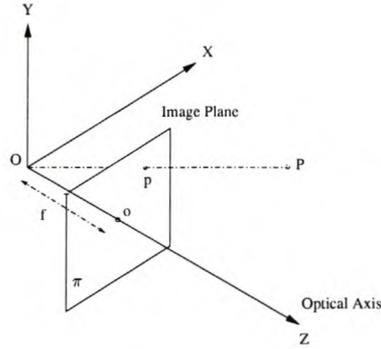
Kruppa's [38] equations relate the epipolar transformation to the image of the absolute conic. Three epipolar transformations, which implies three camera motions, are required to solve for the absolute conic. Thus by calculating the fundamental matrix [24], which defines the epipolar transformation, Kruppa's equations can be used to determine the image of the absolute conic. The absolute conic is related to the intrinsic camera matrix  $A$  which is given in (3.8) as follows:

$$w_\infty = A^{-T} A^{-1} \tag{3.1}$$

The camera matrix can then be solved by using *Cholesky decomposition*.

Most self-calibration algorithms are concerned with unknown but constant intrinsic camera parameters, and examples can be found in [25], [17] and [40].

Self-calibration enables us to obtain a model which makes metric reconstructions possible. However the model is still only determined up to a scale factor, which is not sufficient for many applications requiring precise physical measurements.



**Figure 3.1:** *Pinhole Camera model*

### 3.3 Camera Model

The most common geometric camera model of an intensity camera is the pinhole model shown in figure 3.1. The model consists of a plane  $\pi$ , the image plane, and a 3D point  $\mathbf{O}$ , the center or *focus of projection*. The distance between  $\pi$  and  $\mathbf{O}$  is the focal length  $f$ . The line through  $\mathbf{O}$  and perpendicular to  $\pi$  is the *optical axis*, and  $\mathbf{o}$ , the intersection between  $\pi$  and the optical axis is called the *principal point* or *image center*. As shown in figure 3.1,  $\mathbf{p}$ , the image of  $\mathbf{P}$ , is the point where the straight line through  $\mathbf{P}$  and  $\mathbf{O}$  intersects the image plane  $\pi$ .

Consider the 3D reference frame where  $\mathbf{O}$  is the origin and the plane  $\pi$  is orthogonal to the  $Z$  axis. This defines the *frame of reference* of the camera, also referred to as the camera coordinate system.

#### 3.3.1 Intrinsic Parameters

We will now describe all the intrinsic parameters in detail, followed by the extrinsic parameters in the next section.

##### Focal Length

Let  $\mathbf{P} = [X, Y, Z]^T$  be a point in camera coordinates and  $\mathbf{p} = [x, y, z]^T$  be its projection respectively. The principal equations of the perspective transform is then:

$$\begin{aligned} x &= f \frac{X}{Z} \\ y &= f \frac{Y}{Z} \\ z &= f \end{aligned}$$

(3.2)

where  $f$  is the camera focal length in millimetres, as shown in figure 3.1. Note that this transformation is the basis of the perspective camera model. The vector  $\mathbf{p}$  is defined in the *retinal coordinate system*, which implies that it lies on the image plane. With these vectors we often write  $\mathbf{p} = [x, y]^T$  instead of  $\mathbf{p} = [x, y, z]^T$ , since we know that  $z = f$ .

### Image Center

Recall now that  $o$  is called the image center or *principal point*, and is defined as the point where the optical axis intersects the image plane,  $\pi$ . We define the image center,  $C_x$  and  $C_y$ , to be the image coordinates of the principal point, which coincides with the center of radial distortion. In an ideal lens system the optical axis is defined as the straight line passing through all of the radii of curvature of the lens elements. The rotational symmetry of the system naturally leads to imaging properties that are radially symmetric around the optical axis. In a real lens the modelling is not so simple due to manufacturing methods. Ideally the optical axis would coincide with the mechanical axis, but practically this is rarely the case. The difference is called *decentration*. Willson [37] defines 16 different image centers along with explanations of their applicability. The most important fixed-lens center is the *center of radial distortion*, which models the radial distortion effect along with the distortion coefficients. Our model assumes that this center of distortion coincides with the *principal point*, and is represented by  $C_x$  and  $C_y$ , which is an approximation since the principal point and center of distortion do not necessarily coincide.

### Radial Distortion

Various distortions occur in the image due to nonlinear effects of the lenses. These are most notably radial and tangential distortion. Researchers have found that radial distortion dominates and tangential distortion can normally be neglected [35] [41]. Radial distortion is sometimes referred to as the *fish-eye effect*, and can most clearly be seen along the outer edges of images where straight lines appear to be curved. This effect is due to the varying thickness of the lens resulting in different light diffractions.

Fortunately this distortion can be modelled rather accurately as follows:

$$\begin{aligned} x_u &= x_d(1 + k_1r^2 + k_2r^4 + \dots) \\ y_u &= y_d(1 + k_1r^2 + k_2r^4 + \dots) \\ r^2 &= x_d^2 + y_d^2 \end{aligned} \tag{3.3}$$

where  $(x_d, y_d)$  are the distorted retinal coordinates and  $(x_u, y_u)$  the undistorted retinal coordinates. It has been shown [35],[33] that the first coefficient  $k_1$  of the series in (3.3) is sufficient for accurate modelling of the camera. This is the only parameter which we

use and our experience confirms that it accurately models and corrects for the radial distortion.

Note that even though the conversion from distorted to undistorted radial coordinates is straightforward as shown in (3.3), the reverse transformation requires solving a  $3^d$  order polynomial.

### Scaling Factor

The horizontal scale factor for CCD cameras is the adjustment factor for the transform from the retinal plane to the computer image. Since the actual spacing between adjacent CCD elements are supplied by the camera manufacturer, the transformation can be achieved without knowing the horizontal scale factor. The vertical scale factor needs no calibration since there is a one-to-one correspondence between the CCD elements and the image plane due to the scanning process timing signals which are in PAL format.

The uncertainty of the horizontal scale factor comes from the imperfect match between the computer image acquisition hardware and the camera hardware. During scanning, spatially discrete signals picked up by each row of the sensor array are first converted to an analog waveform, which is then sampled by the computer acquisition hardware into a number of spatially discrete samples and put into a frame buffer. Obviously, the scale factor is the ratio between the number of sensor elements in a row of the sensor array (CCD) and the number of pixels in a row in the image buffer. This can be determined beforehand using data supplied by manufacturers and is most often exactly unity, but due to timing signal mismatches the value still needs to be calibrated and could easily have an horizontal effect of 20 pixels [21].

We denote this horizontal scaling factor by  $s_x$ , and it's calibration will be shown later in this chapter.

### CCD Dimensions

The CCD dimensions are merely the physical dimensions of the CCD element which is mostly supplied by the camera manufacturers. We are interested in four parameters here:

- $d_x$  Width of single CCD element in millimetres.
- $d_y$  Height of a single CCD element in millimetres.
- $N_{cx}$  Number of CCD elements in a horizontal scan line.
- $N_{fx}$  Number of frame buffer elements in a horizontal scan line.

The last parameter is actually an attribute of the frame grabber, and not the camera itself. It is included here since it is always being used in conjunction with  $N_{cx}$ .

### 3.3.2 Extrinsic Parameters

The camera reference frame has been introduced for the purpose of writing the fundamental equations of the perspective projection (3.2) in a simplified form. However, the camera reference frame is often unknown, and a common problem is to determine the camera reference frame with respect to some known reference frame using only image information. This problem is often referred to as *pose estimation*. The extrinsic parameters are defined as *any set of geometric parameters that identify uniquely the transformation between the unknown camera reference frame and a known reference frame, called the world reference frame* [33].

A typical choice of representing the extrinsic parameters are:

- A 3x3 orthogonal rotation matrix,  $R$ , that aligns the corresponding axes of the reference frames.
- A 3D translation vector,  $\mathbf{T}$  which describes the relative positions of the reference frames.

The computer vision world is divided on whether the rotation or translation is applied first. We have chosen the rotation as first step to simplify our calibration algorithm, but the choice makes no difference to the final results. It only effects the form of the equations used.

The Euclidean transformation of a point,  $\mathbf{P}_w = [X_w, Y_w, Z_w]^T$  given in world coordinates into a point  $\mathbf{P}_c = [X_c, Y_c, Z_c]^T$  in camera centred coordinates is given by

$$\begin{bmatrix} X_c \\ Y_c \\ Z_c \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ Z_w \end{bmatrix} + \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \quad (3.4)$$

where  $R$  and  $\mathbf{T}$  define the rotation and translation respectively from world coordinates to camera coordinates.

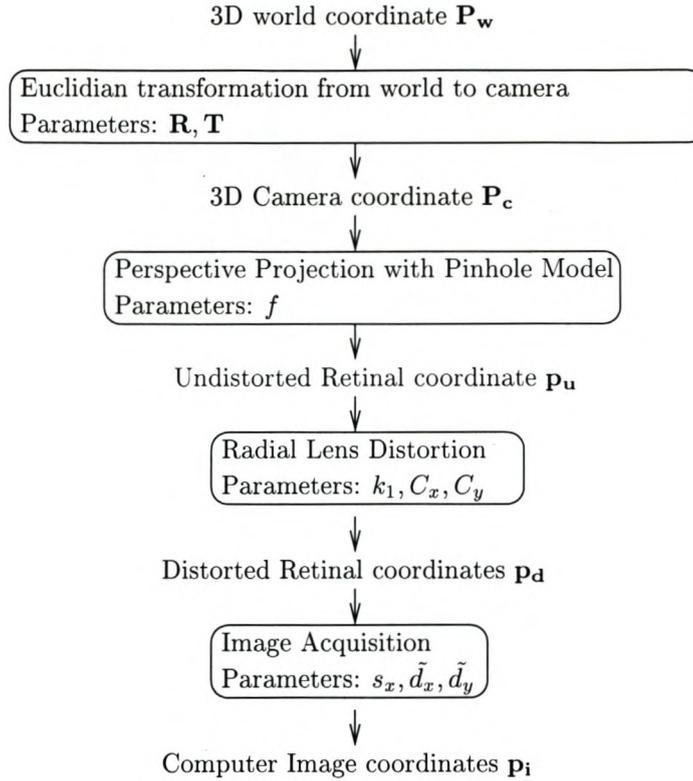
### 3.3.3 Full Mathematical Model

Using all the parameters introduced above it is possible to relate the 3D position of a point in space to it's image on the image plane acquired by the frame grabber. The basic steps are shown in figure 3.2.

Given a point  $\mathbf{P}_w = [X_w, Y_w, Z_w]^T$  in world coordinates, the steps are as follows:

1. World coordinates to camera reference:

$$\mathbf{P}_c = R\mathbf{P}_w + \mathbf{T} \quad (3.5)$$



**Figure 3.2:** 3D Coordinate to Image transformation

2. Apply the perspective equations:

$$\begin{aligned} x_u &= f \frac{X_c}{Z_c} \\ y_u &= f \frac{Y_c}{Z_c} \end{aligned} \tag{3.6}$$

Transform these undistorted retinal coordinates,  $(x_u, y_u)$  to distorted retinal coordinates by solving the following:

$$\begin{aligned} x_u &= x_d(1 + k_1 r^2) \\ y_u &= y_d(1 + k_1 r^2) \\ r^2 &= x_d^2 + y_d^2 \end{aligned} \tag{3.7}$$

The image coordinates are then obtained from

$$\begin{bmatrix} x_{im} \\ y_{im} \\ 1 \end{bmatrix} = \begin{bmatrix} s_x \tilde{d}_x^{-1} & 0 & C_x \\ 0 & \tilde{d}_y^{-1} & C_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_d \\ y_d \\ 1 \end{bmatrix} \tag{3.8}$$

where  $\tilde{d}_x^{-1}$  and  $\tilde{d}_y^{-1}$  are given by:

$$\begin{aligned}\tilde{d}_x^{-1} &= \frac{N_{fx}}{d_x N_{cx}} \\ \tilde{d}_y^{-1} &= \frac{1}{d_y}\end{aligned}\tag{3.9}$$

and  $s_x$  is the horizontal scaling factor discussed above.

If the effect of radial distortion is neglected, the above equations can be combined into a single *projection matrix*,  $P$ , which encodes the extrinsic and intrinsic parameters [33]. The mapping from 3D coordinates to 2D coordinates is a perspective projection, which is represented by a linear transformation in homogeneous coordinates. The projection matrix is defined such that

$$\lambda \mathbf{m}_{\text{im}} = P \mathbf{M}_{\text{w}}\tag{3.10}$$

where  $\lambda$  is an arbitrary scale factor,  $\mathbf{m}_{\text{im}}$  the homogeneous image coordinate and  $\mathbf{M}_{\text{w}}$  the homogeneous world coordinate. As such we define  $P$  as

$$P = A[R|\mathbf{T}]\tag{3.11}$$

with  $R$  and  $\mathbf{T}$  the extrinsic parameters (3.4) and  $A$  given by

$$A = \begin{bmatrix} \frac{fs_x}{d_x} & 0 & C_x \\ 0 & \frac{f}{d_y} & C_y \\ 0 & 0 & 1 \end{bmatrix}\tag{3.12}$$

### 3.4 Intrinsic Calibration Algorithm

The intrinsic calibration is based on the noncoplanar routine first proposed by Tsai [35]. We use the noncoplanar technique, since this allows for the calibration of the horizontal uncertainty factor,  $s_x$ .

Calibration is made possible by a single view of a noncoplanar object, where the image coordinates of at least 7 features are known, as well as the physical dimensions of the object. This routine extracts both intrinsic and extrinsic parameters. We give a short algorithmic overview of the method. Refer to [35] for a detailed analysis.

#### Acquire Features

The first step is to grab an image of the noncoplanar object, and accurately locate the coordinates of the known physical features. We do this via the algorithm described in chapter 2. We denote these feature coordinates as  $(X_{fi}, Y_{fi})$ , where  $i = 1 \dots N$ , with  $N$  the number of features.

### Compute the Distorted Retinal Coordinates

The distorted retinal coordinates, denoted by  $(X_{di}, Y_{di})$  are calculated using the inverse of (3.8), as follows:

$$\begin{aligned} X_{di} &= s_x^{-1} \bar{d}_x (X_{fi} - C_x) \\ Y_{di} &= d_y (Y_{fi} - C_y) \end{aligned} \quad (3.13)$$

for  $i = 1 \dots N$ , where  $\bar{d}_x$  is given by

$$\bar{d}_x = d_x \frac{N_{cx}}{N_{fx}} \quad (3.14)$$

In this stage of the calibration we do not include the image center,  $(C_x, C_y)$ . It is assumed to be exactly in the middle of the image, and experiments by Tsai [35] has shown that this does not significantly affect 3D precision. The image center calibration is added as an extra step to the calibration once all the other parameters have been determined. The horizontal scaling factor,  $s_x$ , is also assumed to be exactly one. This is also determined in a later step. Recall that the parameters  $N_{cx}$  and  $N_{fx}$  are supplied by the CCD camera manufacturer.

### Compute the seven unknowns

The calibration method proposed by Tsai is based on the radial alignment constraint, which basically assumes that distortion is purely radial. This allows us to conclude that, no matter how much distortion, the direction of the vector from the image center to a point stays constant. For the derivation of this constraint, refer to the article by Tsai [35].

The radial alignment constraint allows us to set up the following set of linear equations.

$$\begin{bmatrix} y_{d1} X_{w1} & \dots & y_{dn} X_{wn} \\ y_{d1} Y_{w1} & \dots & y_{dn} Y_{wn} \\ y_{d1} Z_{w1} & \dots & y_{dn} Z_{wn} \\ y_{d1} & \dots & y_{dn} \\ -x_{d1} X_{w1} & \dots & -x_{dn} X_{wn} \\ -x_{d1} Y_{w1} & \dots & -x_{dn} Y_{wn} \\ -x_{d1} Z_{w1} & \dots & -x_{dn} Z_{wn} \end{bmatrix}^T \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \alpha_4 \\ \alpha_5 \\ \alpha_6 \\ \alpha_7 \end{bmatrix} = \begin{bmatrix} x_{d1} \\ \vdots \\ x_{dn} \end{bmatrix} \quad (3.15)$$

Where  $\alpha_i$  for  $i = 1..7$  is defined as

$$\begin{aligned}
 \alpha_1 &= s_x r_1 T_y^{-1} \\
 \alpha_2 &= s_x r_2 T_y^{-1} \\
 \alpha_3 &= s_x r_3 T_y^{-1} \\
 \alpha_4 &= s_x T_x T_y^{-1} \\
 \alpha_5 &= r_4 T_y^{-1} \\
 \alpha_6 &= r_5 T_y^{-1} \\
 \alpha_7 &= r_6 T_y^{-1}
 \end{aligned}
 \tag{3.16}$$

and  $(X_{wi}, Y_{wi}, Z_{wi})$  are the known world coordinates of the detected features, in the calibration object's coordinate system.

The elements of the rotation matrix,  $R$ , is defined as

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix}
 \tag{3.17}$$

Since we should have much more than 7 features, this system is overdetermined, and the solution can be obtained by the Moore-Penrose inverse.

We can calculate the value of  $T_y$  by exploiting the orthonormality of the rotation matrix,  $R$ . One implication of the orthonormality is that

$$r_4^2 + r_5^2 + r_6^2 = 1
 \tag{3.18}$$

and thus we can state that

$$|T_y| = [\alpha_5^2 + \alpha_6^2 + \alpha_7^2]^{-\frac{1}{2}}
 \tag{3.19}$$

### Determine sign of $T_y$

Eq. 3.19 determines the absolute value of  $T_y$ , but we are still left with the problem of determining the sign. This can be done by projecting a point on the object in world coordinates into the camera coordinate system, and comparing the coordinate signs to the known retinal plane coordinate signs. If there is a sign difference,  $T_y$  will be negative.

Thus we calculate

$$\begin{aligned}
 x &= r_1 x_w + r_2 y_w + r_3 z_w + T_x \\
 y &= r_4 x_w + r_5 y_w + r_6 z_w + T_y
 \end{aligned}
 \tag{3.20}$$

and compare the signs of  $x, y$  with the signs of the known image coordinate, and if there is a difference we negate the value of  $T_y$ .

**Determine horizontal scaling factor  $s_x$** 

Determine the horizontal scaling factor  $s_x$  using the following equation derived from (3.16) and the orthonormality constraint on  $R$ .

$$s_x = \sqrt{[\alpha_1^2 + \alpha_2^2 + \alpha_3^2]|T_y|} \quad (3.21)$$

**Compute the Rotation matrix**

Compute  $r_1, r_2, r_3, r_4, r_5, r_6$  and  $T_x$  from the solution to (3.15) and (3.16). Now the third row of  $R$  can be calculated as the cross-product of the first two rows, using the orthonormal property of  $R$  and the right-handed rule. Thus

$$R = \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_2r_6 - r_3r_5 & r_3r_4 - r_1r_6 & r_1r_5 - r_2r_4 \end{bmatrix} \quad (3.22)$$

**Compute  $f$  and  $T_z$** 

By ignoring radial distortion it is possible to linearly compute  $f$  and  $T_z$  by solving the following overdetermined linear system.

$$\begin{bmatrix} r_4X_{w_1} + r_5Y_{w_1} + r_6Z_{w_1} + T_y & -y_{d_1} \\ \vdots & \vdots \\ r_4X_{w_n} + r_5Y_{w_n} + r_6Z_{w_n} + T_y & -y_{d_n} \end{bmatrix} \begin{bmatrix} f \\ T_z \end{bmatrix} = \begin{bmatrix} (r_7X_{w_1} + r_8Y_{w_1} + r_9Z_{w_1})x_{d_1} \\ \vdots \\ (r_7X_{w_n} + r_8Y_{w_n} + r_9Z_{w_n})x_{d_n} \end{bmatrix} \quad (3.23)$$

Solving this set of linear equations by means of the Moore-Penrose inverse gives an initial estimate of  $f$  and  $T_z$  which will be exact in the case of a ideal camera with no lens distortion. The next step is to determine the first order radial distortion coefficient,  $k_1$ , along with the exact values for  $f$  and  $T_z$ .

**3.4.1 Nonlinear Optimization**

The method proposed by Tsai used nonlinear optimization to solve for the distortion coefficient and image center,  $[k_1, C_x, C_y]$ , as well as to improve the general model. There has been several new methods proposed to determine the image center as well as the distortion coefficient, and most of these are based on detecting parallel lines in the image [21]. However the technique used here was shown to be very simple to implement as well as accurate [6].

For our nonlinear optimization operations we use a multi-dimensional unconstrained optimization technique called *Nelder-Mead Simplex search*. We also implemented the routines using a *Levenberg-Marquadt* optimiser, achieving the same results as the Nelder-Mead search. We will now first define our error criterion, after which we will detail our optimization steps.

### Error Criterion

Let  $\mathbf{P}_w = [X_w, Y_w, Z_w]^T$  be the world coordinate of a feature point on the calibration object, and  $\mathbf{P}_{im} = [x_{im}, y_{im}]^T$  be the corresponding image point as found by the corner detector. We compute the reprojection of all the known 3D world coordinates located on the calibration object using the camera model parameters calculated above. This implies that the known 3D world coordinates are projected onto the image using the camera model and the calculated model parameters. The total error is then measured as the mean distance between an image point and its reprojection.

$$\text{Error} = \frac{1}{N} \sum_{i=1}^n ((x_{im_i} - x_{proj_i})^2 + (y_{im_i} - y_{proj_i})^2) \quad (3.24)$$

where  $\mathbf{P}_{proj} = [x_{proj}, y_{proj}]^T$  is the reprojected image coordinate. This gives us an error measure of how accurately our model fits the given training data. This error measure is also consistently used in so called *bundle adjustment* [8] algorithms for camera calibration.

### Determine $k_1$

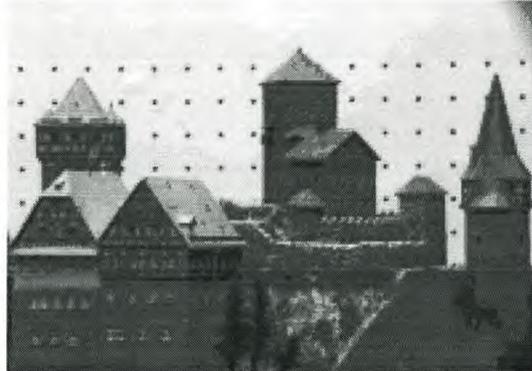
We can now determine the radial distortion coefficient as well as correct for the focal length and  $T_z$ . We perform nonlinear optimization with the free parameters as  $f$ ,  $T_z$  and  $k_1$ , using the error criterion of (3.24). The initial value for  $k_1$  is assumed to be zero, which implies no radial distortion.

### Determine image center ( $C_x, C_y$ )

The image center now needs to be determined. We initiate this step with a global optimization of all the parameters, except the image center. This gives us an good starting position from which to find the image center.

Since the image center and radial distortion are very closely dependant, we perform optimization with those three free parameters, namely  $k_1$ ,  $C_x$  and  $C_y$ . The initial position of the image center is taken as the middle of the image buffer. Once this optimization step is completed, we perform one more global optimization with all the parameters included, which should give us the best possible match using this camera model.

### 3.5 Results



**Figure 3.3:** *CMU image with ground truth*

We used stereo data sets provided by the *Calibrated Imaging Laboratory* [1] of Carnegie Mellon University to demonstrate the accuracy of the algorithm. The scene depicted in figure 3.3 shows a model castle. CMU provides the parameters for the camera which was used to take the images using the same parameter set as Tsai. We are also provided with the image coordinates and real world coordinates of the calibration object in the background, as well as other features in the images. The planar object was moved by an automated jig platform to provide noncoplanar data.

Table 3.1 shows the results provided by CMU, the results of Reg Willson’s [6] implementation of Tsai’s algorithm, Jean-Yves Bouquet’s method, as well as our implementation. The ‘ground-truth’ as supplied by CMU is in fact purely the results of their implementation of Tsai’s algorithm, and thus must be seen as just another calibration effort, and not ground-truth. Jean-Yves Bouquet’s [4] calibration toolbox is based on a paper by Zhang [41], and is also included in a computer vision library released by Intel called OpenCV [2]. The implementation by Bouquet does not use the exact same parameters as our implementation, and some conversion was necessary to be able to compare the methods. The only major difference is that Bouquet implements a radial and tangential distortion model, but he also admits that the distortion is dominated by first order radial distortion, and any additional modelling is unnecessary [41]. We convert his distortion model into a single first order radial distortion coefficient to facilitate comparison.

We use the *mean reprojection error* as an estimate of the accuracy of a given camera model. This entails that the 3D coordinates of the calibration object are projected onto the image using the camera model. These projected features should correspond exactly with the located features, which acted as training data. The mean Euclidean distance in pixels between projected and located features acts as a measure of accuracy, as shown in (3.24).

From table 3.1 it can be seen that all the reported methods perform very similarly.

<i>Parameter</i>	<i>CMU</i>	<i>Willson</i>	<i>Zhang</i>	<i>Our results</i>
$f$	57.439	57.400	57.408	57.374
$C_x$	258.731	258.523	260.177	260.667
$C_y$	200.591	198.472	198.569	198.772
$k_1$	-5.373e-04	-5.259e-04	-5.349e-04	-5.243e-04
$s_x$	1.000241	1.000335	N/A	1.000357
$T_x$	-566.75	-566.60	-567.76	-568.05
$T_y$	-516.25	-514.77	-514.84	-515.01
$T_z$	1767.15	1766.07	1765.67	1764.56
$R_x$	5.244e-04	-3.078e-04	-2.594e-04	1.600e-04
$R_y$	7.636e-03	7.718e-03	7.062e-03	6.886e-03
$R_z$	-1.113e-02	-1.11e-02	-1.113e-02	-1.107e-02
<i>Mean Error</i>	0.124145	0.121025	0.11907	0.1167

**Table 3.1:** Comparison of intrinsic parameters

From the mean error values in the last row it can be seen that the parameters supplied by CMU perform worse than all the other calibration results. The results from our calibration is marginally better than the others. This small difference can be attributed to the non-linear optimiser since we implemented the same algorithm as Willson and CMU.

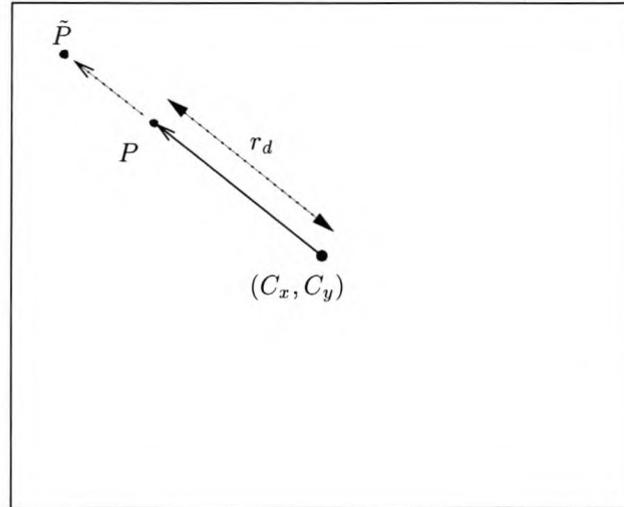
We can see that even though many new calibration methods for single cameras have been proposed since the publication of Tsai's paper [35], no significant advances in accuracy can be observed. The main advantages gained have been flexibility in that methods can use coplanar data as well, and combine multiple views of an object to obtain more accurate results.

### 3.5.1 Radial Distortion

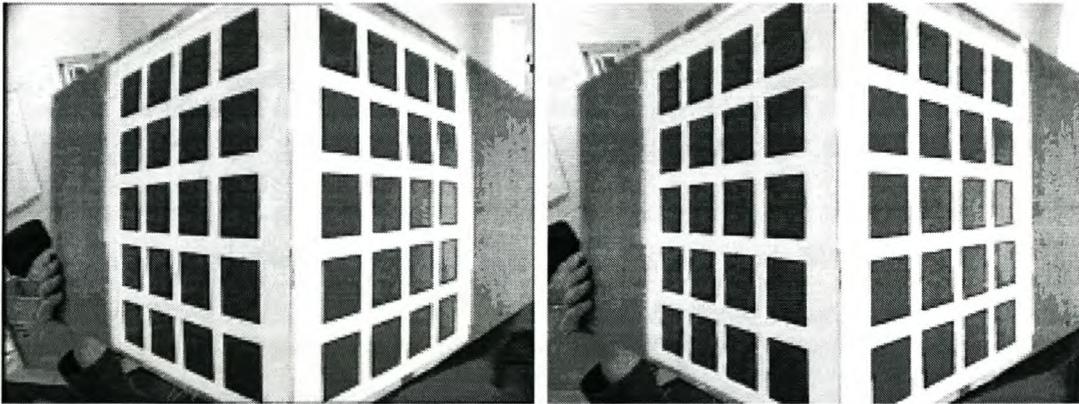
Off-the-shelf CCD cameras usually exhibit significant lens distortion, especially radial distortion. Radial distortion is the effect of imperfect lenses where the magnification is different at the edges than at the center of the image. We model only first order dominant lens distortion, see [35]. It has also been found that any more elaborate modelling would not help since it is negligible compared to sensor quantization, and could also cause numerical instability [36].

Radial distortion can have a major effect on the overall accuracy of a system, and if not modelled accurately it also causes the epipolar calculations to be unstable. In this section we will demonstrate the effect of radial distortion, and how we undistort images.

Figure 3.4 demonstrates the radial distortion effect. The image center is defined as  $(C_x, C_y)$ , with  $\mathbf{P}$  an observed point on the image plane. The true point,  $\tilde{\mathbf{P}}$  is constrained to

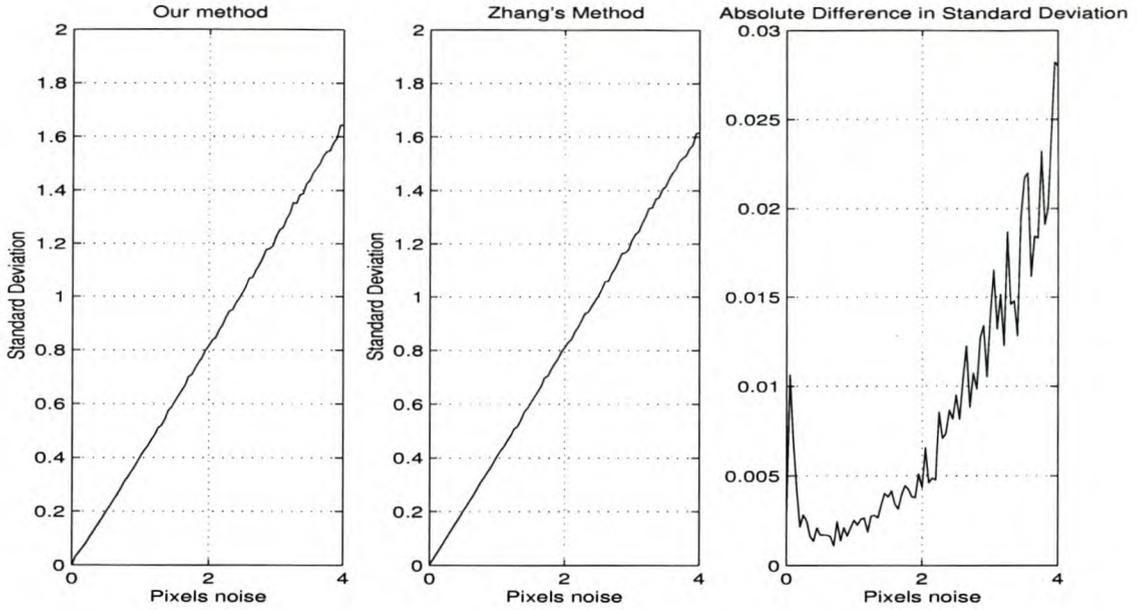


**Figure 3.4:** *Radial Distortion*



**Figure 3.5:** *Radially distorted and undistorted images*

lie along the direction of the vector  $\bar{CP}$ , according to the radial alignment constraint. The position of the new, true point is calculated using (3.7), with  $r_d$  the Euclidean distance from  $C$  to  $P$ . The distortion effect on a feature is thus dependent on its distance from the image center, which is also the center of radial distortion. This can clearly be seen in the left picture of figure 3.5. The lines on the outer edges of the picture are curved towards the center of radial distortion. The image on the right shows the undistorted image, where the lines have been straightened. This undistortion is achieved through individually calculating the undistorted coordinate of each pixel, thus reconstructing the undistorted image.



**Figure 3.6:** Comparison of Single camera calibration methods with Noisy data

### 3.5.2 Robustness against noise

We experimented with how our method as well as Zhang's method of calibration reacts to noisy measurements. In real environments the measurements will always be corrupted by some form of noise, since the components of the system are not ideal. Noise in the system could be from factors such as uneven lighting, CCD blooming and inaccuracies in the feature measurement as well as the calibration object manufacturing.

Our experiment consisted of generating synthetic calibration data of a noncoplanar object with 36 squares on each plane, which supplies 288 feature points in total. This is placed at a distance of approximately 1m from the camera which closely resembles what we would do in practise. We use the camera model described earlier with realistic parameters to project the 3D data onto an image plane.

The image coordinates of this object is then used to perform calibration using both methods. Since it is purely synthetic data with no noise we would expect both methods to perform perfectly, which they do as can be seen in figure 3.6.

We now add random noise with a uniform density to each feature measurement in the image, and perform the calibration again. The measure of error we use is the standard deviation of the distance between the reprojection of a feature and it's measured location. Thus with  $(x_{im_i}, y_{im_i})$  as our image coordinates, and  $(x_{proj_i}, y_{proj_i})$  as our reprojected coordinates, the standard deviation is defined as

$$s = \sqrt{s_x^2 + s_y^2} \quad (3.25)$$

where

$$\begin{aligned}
 s_x &= \left( \frac{1}{n} \sum_{i=1}^n (x - \bar{x})^2 \right)^{\frac{1}{2}} \\
 s_y &= \left( \frac{1}{n} \sum_{i=1}^n (y - \bar{y})^2 \right)^{\frac{1}{2}}
 \end{aligned}
 \tag{3.26}$$

and

$$\begin{aligned}
 x &= x_{im_i} - x_{proj_i} \\
 y &= y_{im_i} - y_{proj_i} \\
 \bar{x} &= \frac{1}{n} \sum_{i=1}^n x_i \\
 \bar{y} &= \frac{1}{n} \sum_{i=1}^n y_i
 \end{aligned}
 \tag{3.27}$$

This process is performed at intervals of 0.05 pixels noise, from 0 pixels up to 4 pixels noise. For each level of noise we perform 10 calibrations and average the results, which are shown in figure 3.6, with the standard deviation of our method on the left, the standard deviation of Zhang's method in the center, and the absolute difference between the two on the right. We display the absolute difference purely to demonstrate how closely the two methods perform, which is almost identical. This can be attributed to the fact that both employ nonlinear optimizers after the linear estimation of the parameters.

We can now see that our calibration method for the intrinsic parameters of the camera performs accurately even in the presence of the worst noise we would expect in a calibration environment.

# Chapter 4

## Stereo Camera Calibration

### 4.1 Introduction

Stereo vision refers to the ability to *infer 3D information of a scene from two images taken from different viewpoints*. The simplest demonstration of the essence of stereo vision is to hold an object in front of your face and alternatively close the left and right eyes. Observe that the relative position of the object and background seems to change. It is exactly this difference in retinal position that is used by the brain to reconstruct a 3D scene. This is the essence of what we try to duplicate using stereo vision.

From a computational point of view the stereo system must solve two problems. The first is that of correspondence, in which a feature in the left image must be related to its corresponding physical feature in the right image. This problem is particularly difficult in areas of little texture and at occlusions. This is a problem in its own right, and will be further discussed in chapter 6.

The second problem is that of 3D reconstruction. With the correspondence problem solved, what does a pair of features tell us about the 3D structure? If we have calibrated our cameras as a stereo pair, we can use *triangulation* to determine the 3D position of a feature pair. Triangulation is a method of projecting rays from our respective cameras through the observed features, and finding the intersection of these rays to determine the 3D coordinates of the physical feature. This will be discussed in chapter 5.

The goal of this chapter is to demonstrate how we achieve the stereo calibration of our cameras. Having done the intrinsic calibration we need to determine the *extrinsic parameters* of the stereo pair, which consists of a *rotation* and *translation* defining the relative position and orientation of the respective cameras. We will explain the epipolar geometry on which the method is based, followed by a detailed derivation of the method. Finally some results will demonstrate the usefulness and robustness of the method.

## 4.2 Overview of our calibration method

We propose a method of calibrating stereo cameras which is based on the calculation of the fundamental matrix. We will give an short explanation to provide a framework for the rest of the chapter where more detail will be given.

The calibration of a stereo rig entails determining the translation,  $\mathbf{T}$  and rotation  $R$  which relates the coordinate systems of the two respective cameras. With the calibration method which was proposed by Tsai, or any other calibration method which focusses on a single camera the procedure would be as follows.

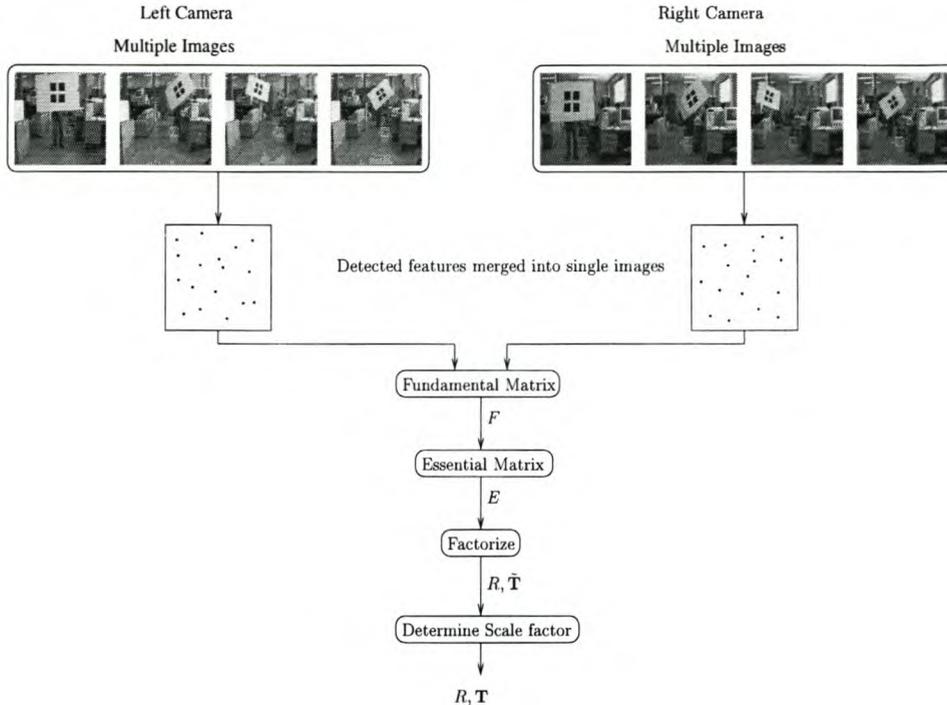
Images are obtained from both cameras at the same instant while they observe the calibration object. This enables us to calibrate both cameras in relation to this object. For the calibration object to be simultaneously visible to both cameras often requires that it be unacceptably far away from the cameras and thus requires a large object to achieve calibration accuracy.

The intrinsic parameters are determined as well as both cameras extrinsic parameters with relation to the object. These are defined as  $(R_l, \mathbf{T}_l)$  and  $(R_r, \mathbf{T}_r)$  for the left and right camera respectively. The extrinsic parameters of the stereo system,  $R$  and  $\mathbf{T}$  can now be calculated as

$$\begin{aligned} R &= R_r R_l^T \\ \mathbf{T} &= \mathbf{T}_l - R^T \mathbf{T}_r \end{aligned} \tag{4.1}$$

This method works if the calibration data is noise-free and very accurate. The moment noise is introduced into the system the accuracy deteriorates significantly, as will be shown later. The main disadvantage of this method is that the cameras are calibrated separately, ignoring the epipolar constraint. Also this method gives accurate results within the area occupied by the calibration object, but performs badly in other areas. This was also found by Chen et al [10] who proposed what they called *virtual objects* to combine multiple views of the object into one big virtual object. This was done to increase the area occupied by the calibration object and thus also the accuracy. Their method was based on a structure-from-motion algorithm to determine camera pose and then combined all the data into one coordinate system. We propose a much simpler method for the case of only two cameras. The broad outline of the algorithm is shown in figure 4.1.

The epipolar geometry, and specifically the fundamental matrix, defines the relationship between points observed in two cameras. If we have eight corresponding features in both images, we can calculate the fundamental matrix, and obtain the extrinsic parameters of the system. However the calculation of the fundamental matrix is very sensitive, and we opt for a least squares solution to the problem using as much input data as available. By acquiring multiple images of the calibration object being moved through the



**Figure 4.1:** *Stereo Calibration algorithm*

scene, we can combine all the data into two images, one for each camera. This simulates using a single calibration object filling most of the scene and thus improves our accuracy over that region. These two composed data sets are then used to calculate the fundamental matrix. This can be reduced to an essential matrix using the intrinsic parameters of the cameras previously obtained. The essential matrix is then factorized to determine for the rotation matrix,  $R$ , as well as the translation vector  $\tilde{\mathbf{T}}$  which is defined up to a scale factor. The scale factor is determined by knowing any physical distance in the image, in our case any measurement on our calibration object.

We now describe in detail the epipolar geometry as well as our calibration method.

### 4.3 Epipolar Geometry

Different views of a scene are not unrelated and several relationships exist between two, three or more views. These are very important for calibration and reconstruction from images and have been extensively studied.

Stereo geometry, also known as *epipolar geometry* is shown in Figure 4.2. The figure shows two pinhole cameras, their projection centers,  $\mathbf{O}_l$  and  $\mathbf{O}_r$ . Each camera defines a 3D reference frame, the origin of which coincides with the projection center, and with the Z-axis along the optical axis. The vectors  $\mathbf{P}_l$  and  $\mathbf{P}_r$  refer to the same 3D point  $\mathbf{P}$ , given in the coordinate systems of the two cameras. The vectors  $\mathbf{p}_l$  and  $\mathbf{p}_r$  refer to the



(4.2) now becomes

$$\mathbf{P}_r^T E \mathbf{P}_l = 0 \quad (4.6)$$

with

$$E = RS \quad (4.7)$$

Using the perspective projection defined in (3.2), (4.6) can be written as

$$\mathbf{p}_r^T E \mathbf{p}_l = 0 \quad (4.8)$$

The matrix  $E$  is called the *essential matrix*, and establishes a relationship between the epipolar constraint and the extrinsic parameters of the stereo system. It is this matrix that allows us to combine multiple views into one set of equations to solve the extrinsic parameters.

Note that these calculations are all performed on the retinal coordinates. Thus in order to use this, the transformation from image coordinates to retinal needs to be known which implies that the intrinsic camera parameters are known. We will generalise the algorithm for the case where the parameters are not known.

## 4.5 Fundamental Matrix

The fundamental matrix is closely related to the essential matrix via the intrinsic parameters of the cameras. We will now show how the fundamental matrix can be calculated directly from image coordinates.

Let  $\tilde{\mathbf{p}}_l$  and  $\tilde{\mathbf{p}}_r$  be the pixel coordinates of vectors  $\mathbf{p}_l$  and  $\mathbf{p}_r$ ,

$$\begin{aligned} \mathbf{p}_l &= M_l^{-1} \tilde{\mathbf{p}}_l \\ \mathbf{p}_r &= M_r^{-1} \tilde{\mathbf{p}}_r \end{aligned} \quad (4.9)$$

where  $M_l$  and  $M_r$  are the intrinsic matrices (3.8) of the left and right camera respectively.

By substituting (4.9) into (4.8) we have

$$\tilde{\mathbf{p}}_r^T F \tilde{\mathbf{p}}_l = 0 \quad (4.10)$$

where

$$F = M_r^{-T} E M_l^{-1} \quad (4.11)$$

$F$  is known as the fundamental matrix and was first proposed by Luong [24]. The essential and fundamental matrices are very similar, the difference being that the fundamental matrix is computed directly from pixel coordinates, and does not require *a priori* knowledge of the intrinsic parameters of the cameras.

Equation (4.10) is linear and homogeneous in the 9 unknown coefficients of  $F$ . Thus we know that if we are given 8 matches we will be able to determine an unique solution to  $F$  up to an unknown scale factor. This approach was introduced by Longuet-Higgins [23] and has been extensively used for the computation of the essential matrix. We use it to solve for the fundamental matrix, and it is referred to as the 8-point algorithm. Written in matrix form, we need to solve

$$\mathbf{p}_i^T \mathbf{f} = 0 \quad (4.12)$$

where

$$\begin{bmatrix} p_{l_{x_1}} p_{r_{x_1}} & \cdots & p_{l_{x_n}} p_{r_{x_n}} \\ p_{l_{y_1}} p_{r_{x_1}} & \cdots & p_{l_{y_n}} p_{r_{x_n}} \\ p_{r_{x_1}} & \cdots & p_{r_{x_n}} \\ p_{l_{x_1}} p_{r_{y_1}} & \cdots & p_{l_{x_n}} p_{r_{y_n}} \\ p_{l_{y_1}} p_{r_{y_1}} & \cdots & p_{l_{y_n}} p_{r_{y_n}} \\ p_{r_{y_1}} & \cdots & p_{r_{y_n}} \\ p_{l_{x_1}} & \cdots & p_{l_{x_n}} \\ p_{l_{y_1}} & \cdots & p_{l_{y_n}} \\ 1 & \cdots & 1 \end{bmatrix}^T \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \\ F_{21} \\ F_{22} \\ F_{23} \\ F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} = 0 \quad (4.13)$$

In practise we have more than 8 points, and the use of a least-squares method is recommended.

$$\min_F \sum_i (\tilde{\mathbf{p}}_r^T F \tilde{\mathbf{p}}_l)^2 \quad (4.14)$$

However if the fundamental matrix is obtained numerically as in (4.14) the rank constraint is not taken into account. This is enforced by a standard SVD method shown later. We decompose the overdetermined system of (4.13) and solve using the singular value decomposition. This linear estimate of  $F$  we use as starting point for a nonlinear optimization.

### 4.5.1 Normalization

Trucco & Verri [33] show that the calculation of the fundamental matrix is very sensitive to noise in the measurements, and that the results are greatly improved by normalization. A simple procedure to avoid numerical instability is to shift the points so that the mean is 0, and scale them to a norm of 1.0.

This scaling is done by two 3x3 matrices as follows,

$$\begin{aligned} \bar{\mathbf{p}}_l &= H_l \tilde{\mathbf{p}}_l \\ \bar{\mathbf{p}}_r &= H_r \tilde{\mathbf{p}}_r \end{aligned} \quad (4.15)$$

where  $\bar{\mathbf{p}}_l, \bar{\mathbf{p}}_r$  are the normalized coordinates,  $\tilde{\mathbf{p}}_l, \tilde{\mathbf{p}}_r$  are the pixel coordinates and

$$H_l = \begin{bmatrix} \frac{\sqrt{2}}{\sigma} & 0.0 & -\frac{\sqrt{2}\bar{x}}{\sigma} \\ 0.0 & \frac{\sqrt{2}-\sqrt{2}\bar{y}}{\sigma} & \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \quad (4.16)$$

where  $\sigma$  is the variance of the coordinate set, and  $\bar{x}, \bar{y}$  are the means of the  $x$  and  $y$  coordinates respectively. The normalization matrix for the right camera,  $H_r$  is calculated in the same way.

The coordinates are all normalized using  $H_l, H_r$ , and then we calculate  $\tilde{F}$  using the above linear method.

$F$  is then obtained as

$$F = H_r^T \tilde{F} H_l \quad (4.17)$$

The disadvantage of the linear method is that the rank-2 constraint is not enforced. Standard practise is to enforce the constraint after the calculation by taking the singular value decomposition of  $F$  and setting the smallest singular value to be zero.

This gives us a good estimate of the fundamental matrix, but it has been shown that the calculation can be further refined by taking into account that  $F$  has only 7 degrees of freedom. This is enforced by the parametrization of  $F$  into 7 parameters.

## 4.5.2 Parametrization

This parametrization of  $F$  was proposed by [24], and is a simple method to derive 7 parameters from  $F$ , perform non-linear optimization on these parameters to constrain the degrees of freedom, and reconstruct the optimized matrix  $F$  from the parameters.

The matrix is parametrized into  $a, b, c, d$  which are four coefficients of a homography between corresponding epipolar lines, the largest of which is used to normalize the system, resulting in only three degrees of freedom for the four parameters. If the left and right epipoles are represented by  $\mathbf{e} = (e_1, e_2)$  and  $\tilde{\mathbf{e}} = (\tilde{e}_1, \tilde{e}_2)$  we define

$$F = \begin{bmatrix} b & a & -ae_2 - be_1 \\ -d & -c & ce_2 + de_1 \\ d\tilde{e}_2 - b\tilde{e}_1 & c\tilde{e}_2 - a\tilde{e}_1 & (ae_2 + be_1)\tilde{e}_1 - (ce_2 + de_1)\tilde{e}_2 \end{bmatrix} \quad (4.18)$$

Inverting (4.18), the parameters of the parametrization are given by

$$\begin{aligned}
 a &= F_{12} \\
 b &= F_{11} \\
 c &= -F_{22} \\
 d &= -F_{21} \\
 e_1 &= \frac{F_{23}F_{12} - F_{22}F_{13}}{F_{22}F_{11} - F_{21}F_{12}} \\
 e_2 &= \frac{F_{13}F_{21} - F_{11}F_{23}}{F_{22}F_{11} - F_{21}F_{12}} \\
 \tilde{e}_1 &= \frac{F_{32}F_{21} - F_{22}F_{13}}{F_{22}F_{11} - F_{21}F_{12}} \\
 \tilde{e}_2 &= \frac{F_{31}F_{12} - F_{22}F_{13}}{F_{22}F_{11} - F_{21}F_{12}}
 \end{aligned} \tag{4.19}$$

Refer to [39] for a detailed analysis of the parametrization.

### 4.5.3 Non Linear Criteria

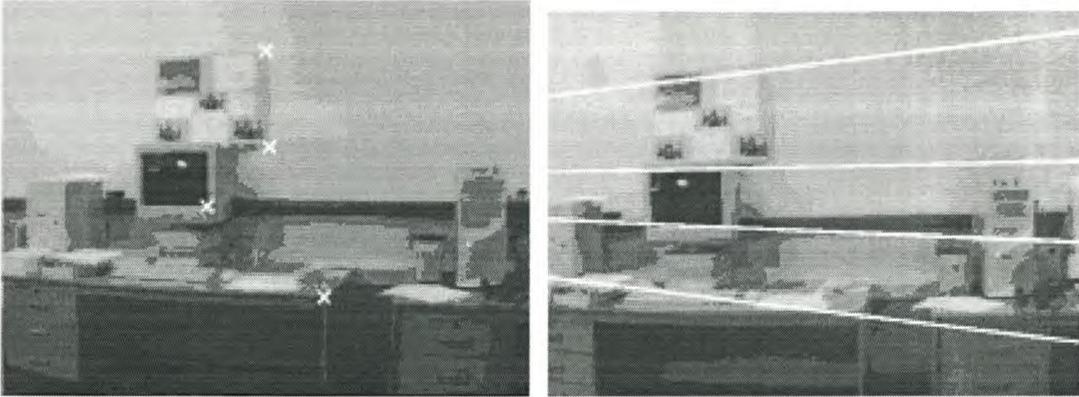
Using the parametrization defined above the accuracy of the fundamental matrix can be further enhanced by non-linear optimization such as the Nelder-Mead Simplex search. The optimization is performed on the fundamental matrix with the following criterion proposed by Luong [24],

$$\min F [d(\tilde{\mathbf{m}}, F\mathbf{m})^2 + d(\mathbf{m}, F^T\tilde{\mathbf{m}})^2] \tag{4.20}$$

where  $d$  is a distance measure in the image plane. This error criterion minimises the distance between points and their corresponding epipolar lines and also ensures that the two images play a symmetric role.

## 4.6 Epipolar Lines

In figure 4.3 we show a pair of stereo images obtained from two cameras with parallel reference frames, i.e. epipolar lines should be parallel. On the left image (taken by the left camera) we marked four features. Their corresponding epipolar lines are displayed on the right image. These lines were plotted using the fundamental matrix computed by the simple 8-point algorithm of (4.13). The feature matches used to compute this was obtained by taking multiple images of our coplanar calibration object. There was no optimization, normalization or parametrization applied. The error criterion (4.20) gives



**Figure 4.3:** *Epipolar Lines*

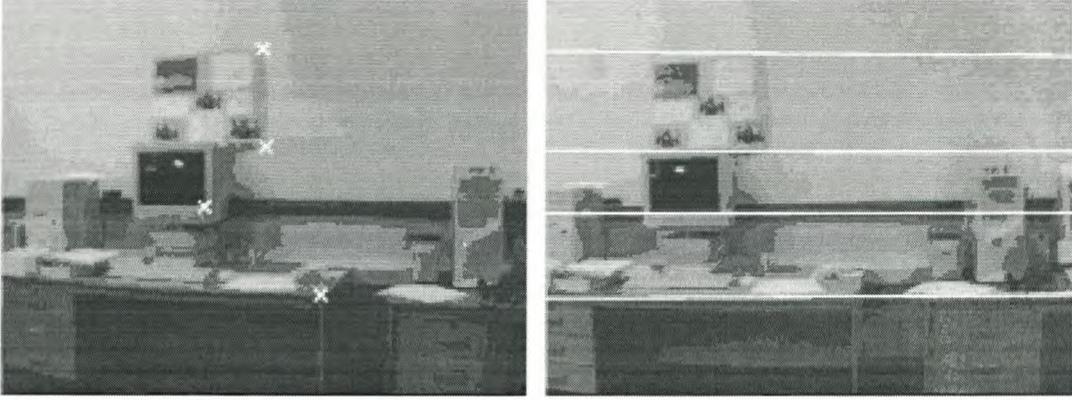
$e = 3.8199$ . From the epipolar lines drawn it seems as if the cameras were not parallel, which in fact they were. Note that the epipolar lines still lie on their corresponding matches, even though their orientation is incorrect. This combined with the error rate shows that the simple 8-point algorithm is not sufficient for accurate calibration.

Figure 4.4 shows the same images, but in this case we calculated the fundamental matrix more accurately. Table 4.1 shows the four error values we obtained with different calculation methods. The first value, denoted 8-point, is the simple overdetermined system of (4.20) solved using a singular value decomposition. The second value is the same algorithm, but with the normalization shown in (4.16) applied. The improvement is remarkable - an order of magnitude. The third value was computed by following the normalization and 8-point algorithm with a non-linear optimiser using the error criterion of (4.16). This result was only slightly better than the second value. For the last value shown we follow the previous answer with a nonlinear optimization of the parametrization shown in (4.18) and (4.19). This also provides a very marginal increase in the system.

<i>Method</i>	<i>8-point</i>	<i>Normalized 8-point</i>	<i>Nonlinear</i>	<i>Parametrization</i>
<i>Error</i>	3.8199	0.37635	0.363112	0.36218

**Table 4.1:** *Fundamental Matrix results*

From this example it can be seen that significant accuracy gains are made by normalising the coordinates before calculation of the fundamental matrix. The non-linear optimization and parametrization make much smaller contributions to accuracy, but can also in some cases play a bigger role.

Figure 4.4: *Epipolar Lines*

## 4.7 Factorization

Once the fundamental matrix  $F$  has been calculated accurately, it can be converted to the essential matrix via (4.11). Taking into account that  $R$  is an orthonormal matrix and  $S$  an antisymmetric matrix, we can factorize  $E$  into rotation and translation. The translation vector is determined up to an arbitrary scale factor. This ambiguity can be solved by knowing any metric distance in the images, which we obtain from any view of our calibration object with its known dimensions.

Let  $\mathbf{c}_i$  and  $\mathbf{r}_i$  for  $i = 1, 2, 3$  be the column vectors of  $E$  and  $R$ . The equation  $E = RS$  (4.7) implies that  $\mathbf{c}_i = \mathbf{t} \times \mathbf{r}_i$  for all  $i$  with  $\mathbf{t}$  the translation vector determined up to a scale factor. The three column vectors of  $E$  are perpendicular to  $\mathbf{t}$  and therefore  $\mathbf{t}$  is parallel to the cross product of any two of the  $\mathbf{c}_i$ 's. Thus the translation vector is simply the cross product [11]:

$$\mathbf{t} = \mathbf{c}_1 \times \mathbf{c}_2 \quad (4.21)$$

where  $\mathbf{t}$  indicates the translation vector up to an unknown scale factor.

The matrix of cofactors  $E^*$  of  $E$  is given by

$$E^* = \begin{bmatrix} \mathbf{c}_2 \times \mathbf{c}_3 & \mathbf{c}_3 \times \mathbf{c}_1 & \mathbf{c}_1 \times \mathbf{c}_2 \end{bmatrix}^T \quad (4.22)$$

which can be rewritten as

$$\begin{aligned} E^* &= \begin{bmatrix} (\mathbf{r}_1^T \mathbf{t}) \mathbf{t} & (\mathbf{r}_2^T \mathbf{t}) \mathbf{t} & (\mathbf{r}_3^T \mathbf{t}) \mathbf{t} \end{bmatrix}^T \\ &= (\mathbf{t} (R^T \mathbf{t})^T)^T \\ &= (\mathbf{t} \mathbf{t}^T R)^T \end{aligned} \quad (4.23)$$

It can then be shown [11] that

$$\begin{aligned} TE &= \mathbf{t}\mathbf{t}^T R - (\mathbf{t}^T \mathbf{t})R \\ (\mathbf{t}^T \mathbf{t})R &= E^{*T} - TE \end{aligned} \tag{4.24}$$

which yields  $R$  as a function of  $E$  and  $\mathbf{t}$ .

## 4.8 Scale Factor

We have now determined, using the fundamental matrix, the extrinsic parameters  $R$  and  $\mathbf{t}$ , where  $\mathbf{t}$  is the translation vector determined up to an unknown scale factor. This scale factor can be determined by knowing any physical measurement in the image. Since we have multiple images of our calibration object, we use the known dimensions on that to determine the scale factor.

This is done by performing the triangulation explained in the next chapter to measure the physical distance between two features on a set of stereo images. The ratio between this measured distance and the true physical distance defines the scale factor, which is then applied to  $\mathbf{t}$  to obtain  $\mathbf{T}$ .

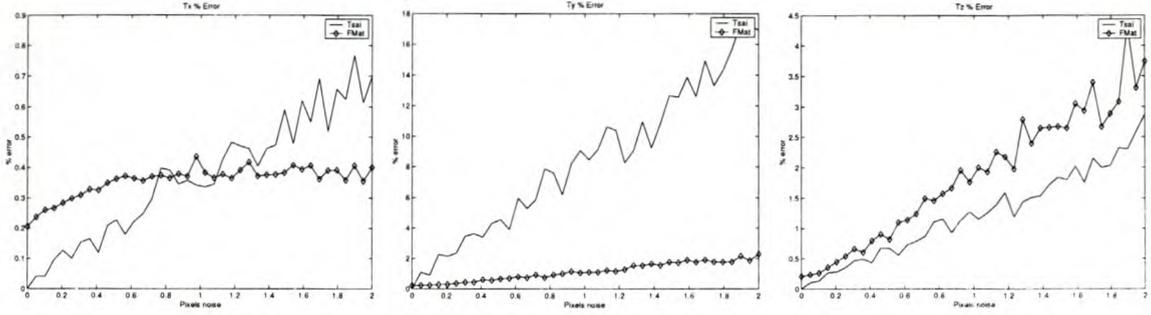
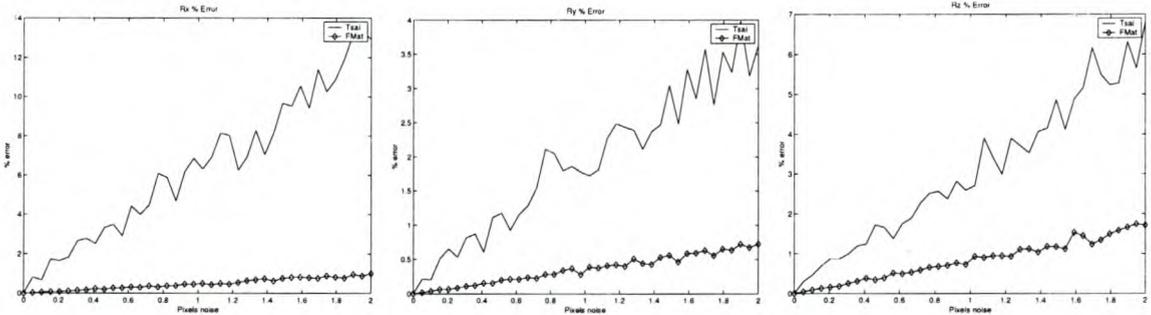
## 4.9 Results

We compared our method of stereo calibration to the more traditional method explained in section 4.2. This was done by generating synthetic data using the pinhole camera model. This consisted of a stereo rig with extrinsic parameters  $\mathbf{T} = [-989.45, 18.73, -18.87]$  and  $R = [-0.026, 0.103, 0.013]$ . The calibration data was four views of a planar object with 64 features on at distances between 1 and 3 metres from the cameras.

The cameras' intrinsic parameters are already known, and the coplanar data was used to determine the extrinsic parameters of the rig. We used both Tsai's [35] algorithm as well as our fundamental matrix algorithm to calculate the extrinsic parameters. As previously, we experimented with noisy measurements by adding uniformly distributed noise with a maximum level between 0 and 2 to each feature coordinate. For each level of noise a 100 calibrations were performed and the results averaged.

In figure 4.5 we plot the percentage errors of the translation vector elements for both methods. The horizontal axis is the level of noise added to each feature in pixels, and the vertical axis is the percentage of deviation from the ground truth. Tsai's method is denoted by the solid line and our method, labelled FMat is denoted by the diamond line.

For the  $T_x$  element of the translation our method starts off a bit worse than Tsai's, but levels off where Tsai's methods error increases. It is worthwhile to note that the error

Figure 4.5: *Translation Vector errors*Figure 4.6: *Rotation Angle errors*

here for both methods is still under 0.8%, which translates to an error less than 8mm on a distance of one meter. In the  $T_y$  case our method is clearly less sensitive to noise, and at a noise level of 2 pixels per feature, our method's error is an order less than Tsai's. The  $Z$  element of translation is the only case where Tsai's method performs consistently better than ours, performing about half a percentage better. The reason for these inconsistencies in sensitivity still has to be investigated in more detail.

Figure 4.6 shows the percentage errors in the rotation angles which are measured in radians. For all three rotation angles our method exhibits far less sensitivity to noise than Tsai's.

We conclude that the method we proposed performs on par with other calibration methods, and in some cases shows far less sensitivity to noisy measurements. The big advantage is that no specific calibration object is needed, merely a set of matches between two images. This can be obtained with multiple views of any easily trackable object. It must also be noted that only 8 matches are needed to calibrate, but greater accuracy can be gained by simply using more input data, which makes the algorithm highly scalable.

# Chapter 5

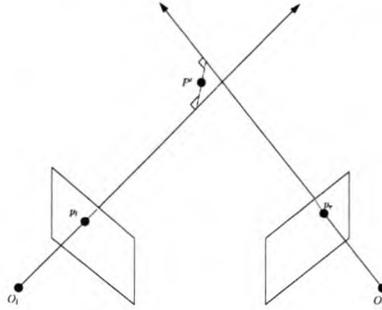
## Triangulation

Triangulation is the process of determining the 3D position of a feature, using two corresponding image features. There are many possible methods of doing this, and we will explain the method we chose as well as our motivation for doing so.

### 5.1 Introduction

We suppose that a point  $\mathbf{Q}$  in  $R^3$  is visible in two images. The two camera projection matrices,  $P$  and  $P'$ , corresponding to the two images are known. Let  $\mathbf{q}$  and  $\mathbf{q}'$  be the projections of  $\mathbf{Q}$  on the left and right camera respectively. From this information we can, for each camera, construct a ray in space from the camera origin through the observed point into space. The triangulation problem is to find the intersection of these two rays in space, which defines the 3D position of the observed point,  $\mathbf{Q}$ . At first this seems like a trivial problem, since intersecting two lines in space is straightforward. Unfortunately, the problem is complicated in a number of ways. The presence of noise in the measurements and inaccuracies in the camera model often cause the lines not to intersect in space. In this case we need to find the best solution to the posed problem. Radial lens distortion can also cause havoc with reconstructions, and this needs to be taken into account. Researchers have proposed many possible ways of solving the triangulation problem, but there is still no consensus on which method works best.

The triangulation problem can be posed in three different scenarios, depending on what information is available. If *both intrinsic and extrinsic* parameters are known we can simply find the intersection of the two rays in space. In the presence of noise we try to find the best solution to this problem. If *only intrinsic parameters* are known we can still produce a reconstruction, albeit only up to a scale factor. This is achieved by using the *essential matrix to estimate the extrinsic parameters*. If we have *no information on any parameters* we can generate a reconstruction only up to an *unknown projective transformation*.



**Figure 5.1:** *Triangulation with non-intersecting rays*

The case we have to deal with is the first where we have complete, accurate models of our cameras, and thus triangulation is a straightforward problem. Triangulation is a small cog in the stereo vision machine, and as such is often overlooked in its importance. Without proper triangulation, 3D reconstructions would be inaccurate no matter how well the rest of the system performs.

## 5.2 Mid-Point Method

A commonly proposed method of triangulation is the mid-point method [33]. The only important point to note is that this method is only suitable when the entire camera model is known, since it is neither affine nor projective-invariant [16]. This is because orthogonality is not an affine concept, and mid-point not a projective concept. Due to these factors many researchers [16] have ignored the usefulness of the mid-point algorithm. With calibrated cameras, it provides an easy-to-implement, accurate and extremely fast method of triangulation.

Shown in figure 5.1 are two cameras, with their respective centers,  $\mathbf{O}_l$  and  $\mathbf{O}_r$ . From each camera a vector is projected through an observed point on the retinal plane,  $\mathbf{p}_l$  and  $\mathbf{p}_r$  respectively. In the absence of measurement noise, inaccuracies in the camera models and discretization of the image, the two vectors would intersect at the point  $\mathbf{P}$ , which corresponds to the 3D position we are observing. However, due to the above mentioned factors, the two vectors do not intersect, and we have to find the best possible solution. The mid-point method attempts this, and work as follows:

Let  $a\mathbf{p}_l (a \in \mathcal{R})$  be the ray  $l$ , through  $\mathbf{O}_l$  and  $\mathbf{p}_l$ . Now let  $\mathbf{T} + bR^T\mathbf{p}_r (b \in \mathcal{R})$  be the ray  $r$ , through  $\mathbf{O}_r$  and  $\mathbf{p}_r$ . Our problem reduces to finding the midpoint,  $\mathbf{P}'$  of the shortest section of the vector,  $\mathbf{w}$ , which is orthogonal to both  $l$  and  $r$ . This is straight forward because the end points of the segment  $\mathbf{w}$  are:

$$\begin{aligned} \mathbf{w}_l &= a_0 \mathbf{p}_l \\ \mathbf{w}_r &= \mathbf{T} + b_0 R^T \mathbf{p}_r \end{aligned} \tag{5.1}$$

where  $\mathbf{w}_l$  and  $\mathbf{w}_r$  denote the end points on the  $l$  and  $r$  vectors respectively,  $R$  and  $\mathbf{T}$  denote the extrinsic parameters of the stereo setup, and  $a_0$  and  $b_0$  are solved from the following linear equation:

$$a_0 \mathbf{p}_l - b_0 R^T \mathbf{p}_r + c_0 (\mathbf{p}_l \times R^T \mathbf{p}_r) = \mathbf{T} \tag{5.2}$$

The midpoint of the segment  $\mathbf{w}$  is then simply determined as the average of the two end points defined in (5.1).

### 5.3 Results

We will give some results demonstrating the accuracy which can be expected from the mid-point method. Using a stereo rig with a baseline of 1 meter, and realistic intrinsic parameters, we generated synthetic data. This consisted of 100 random points distributed in front of both cameras, at a distance of between one and three meters. These points were projected onto the image planes of both cameras to supply the data necessary to perform triangulation. As in the previous chapters we corrupt these pixel coordinates with varying levels of uniformly distributed noise. These noisy features are then used as input to the mid-point triangulation method to determine the 3D coordinates.

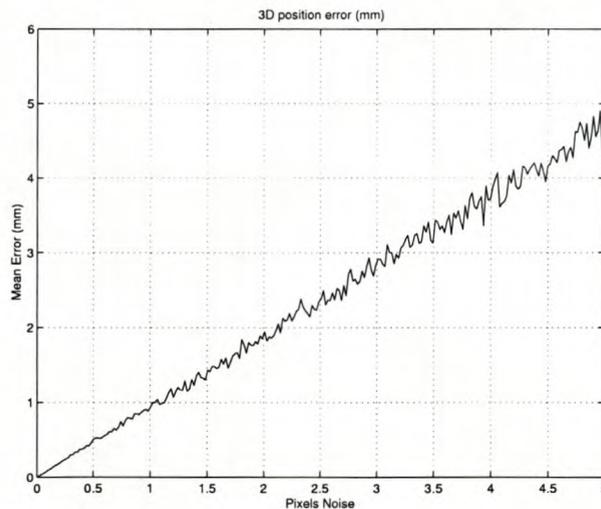
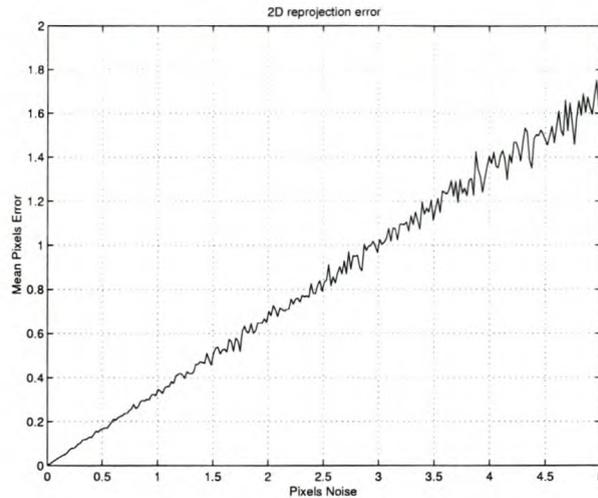


Figure 5.2: 3D position errors



**Figure 5.3:** *2D projection errors*

In the case of no noise these 3D coordinates should correspond exactly to the ground truth coordinates used to generate the image features. With the addition of noise these begin to differ from each other, and it is this difference we show in figure 5.2. This graph shows the mean difference in millimetres between the ground truth position of a feature in 3D space, and the position found by the triangulation method using noisy measurements. The horizontal axis shows the level of noise added, and the vertical axis the error in millimetres. It can be seen that even with fairly severe noise the results are still relatively accurate.

We investigated one other error measure using the same data and results. The 3D positions found by our triangulation are now projected back onto the images using the known camera models. The difference between these *reprojected* coordinates and the coordinates used for the triangulation is now taken as an error measure. This is shown in figure 5.3, where it can be seen that at high noise levels we still get a reasonably accurate reprojection of the 3D features.

We conclude that the mid-point method is well suited to our applications, since we have the entire camera model known. It is extremely fast to calculate which enables us to implement this realtime. It also has no significant problems with noisy measurements.

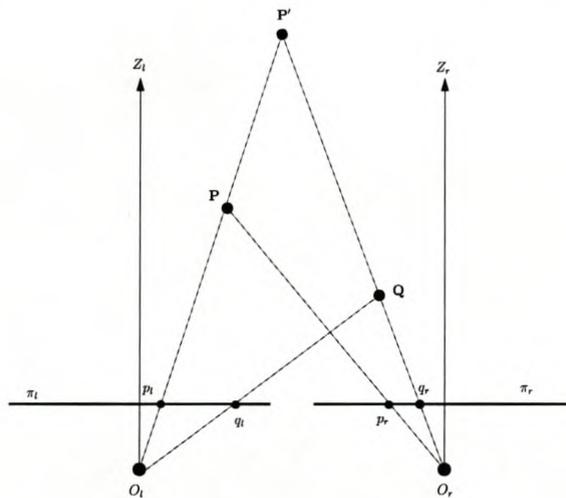
# Chapter 6

## Disparity

### 6.1 Introduction

As stated in chapter 1, the stereo problem consists roughly of two problems. The first being *correspondence*, and the second *reconstruction*. Reconstruction has been dealt with in chapter 5, and now we will turn our attention to the *correspondence* problem.

The correspondence problem entails that for a feature in the left image we need to find its *corresponding* feature in the right image. This correspondence, in relation with the camera models, enable us to do accurate 3D reconstructions.



**Figure 6.1:** *Correspondence problem*

Figure 6.1 shows a very basic top view of a stereo camera setup. The two cameras are denoted by  $O_l$  and  $O_r$ , with their optical axes indicated by the arrows labelled  $Z_l$  and  $Z_r$  respectively. The cameras are observing two points in space,  $\mathbf{P}$  and  $\mathbf{Q}$ , with their projections in the left image labelled  $\mathbf{p}_l, \mathbf{q}_l$  and in the right image  $\mathbf{p}_r, \mathbf{q}_r$ . We can now demonstrate the crucial problem in correspondence. If we choose  $(\mathbf{p}_l, \mathbf{p}_r)$  as matches, the

triangulation will return point  $\mathbf{P}$  as result, but if we choose  $(p_l, q_r)$  as corresponding pair, the triangulation will return point  $\mathbf{P}'$  as answer.

From the triangulation point of view both results are equally valid, showing that everything downstream of the correspondence in the system depends crucially on this stage. The difference in position between the left and right match is referred to as the *disparity*, which can be a two-dimensional vector, but is often reduced to a one-dimensional value by means of *epipolar rectification*, which shall be dealt with later.

In any application where we wish to perform a 3D reconstruction of a scene, we need to establish correspondences between the left and right images. This chapter will detail how we determine dense disparity maps from two images.

## 6.2 Correspondence problem

The correspondence problem can be defined as follows:

“Given an element in the left image, determine the corresponding element in the right image.”

In order to obtain correspondences between the two images, we make some assumptions, namely:

- Most scene points are visible in both images
- Corresponding image regions are similar.
- The cameras are similarly aligned. (both look at the same scene)

All these assumptions hold for a standard *small-baseline* stereo system, in which the distance to the *fixation point* is considerably more than the baseline. The fixation point is defined as the point where the optical axes of the cameras intersect. In the case of parallel cameras this is at infinity. Once these assumptions have been accepted, we need to decide which image elements to match, as well as what similarity measure to use.

We classify correspondence algorithms in two separate classes, *feature based* and *correlation based* algorithms. Although these seem the same from a conceptual point of view, they lead to very different implementations. Feature based algorithms use only specific features to achieve matches, and return a sparse depth map. Correlation based methods attempt to determine a match for every pixel in the image, and thus return dense depth maps. We will now discuss these two classes separately.

### 6.2.1 Feature based correspondence

Feature based correspondence algorithms restrict the search to a sparse set of features. Typically some type of suitable feature descriptor would be chosen, such as lines for an indoor scene. The algorithm then attempts to match these features between the left and right images. Instead of correlation measures they would use distance measures between two features. In this way they would obtain a sparse set of correspondences between the two images. Typical features used are edges, corners, and lines.

The choice of feature descriptor depends strongly on the application. Indoor scenes are perfect locations for feature descriptors, since they all tend to be man-made features. The choice would also depend on factors like lighting conditions, image contrast and speed of execution.

It would seem that correlation based methods with their dense disparity maps are always a better choice, but feature based methods have certain advantages. Feature based methods are in general much faster to process, depending on the feature descriptor used. Also the fact that feature based methods return sparse disparity maps can in some cases not pose any problems. For example indoor navigation often only needs the outlines of objects, and in such a case dense disparity maps are unnecessary.

### 6.2.2 Correlation based Technique

In correlation based techniques, the elements to match are *image windows*, small square windows extracted from the image. The similarity measure used is then some type of correlation. The corresponding element is given by the window that maximises the similarity criterion within a search region.

Given left and right input images,  $I_1$  and  $I_2$ , we wish to extract correspondences. Let  $\mathbf{p}_l$  and  $\mathbf{p}_r$  be pixels in the left and right image,  $2W+1$  the width of the correlation window,  $R(\mathbf{p}_l)$  the search window in the right image corresponding to  $\mathbf{p}_l$ , and  $\psi$  a function of two pixel values,  $(u, v)$ .

Now if we wish to obtain the correspondence of a pixel  $\mathbf{p}_l = [i, j]^T$  in the left image we compute, for each displacement  $\mathbf{d} = [d_1, d_2]^T \in R(\mathbf{p}_l)$ , the following:

$$c(\mathbf{d}) = \sum_{k=-W}^W \sum_{l=-W}^W \psi(I_l(i+k, j+l), I_r(i+k+d_1, j+l+d_2)) \quad (6.1)$$

where the disparity of  $\mathbf{p}_l$  is the vector  $\bar{\mathbf{d}} = [\bar{d}_1, \bar{d}_2]^T$  that maximises  $c(\mathbf{d})$  over  $R(\mathbf{p}_l)$ :

$$\bar{\mathbf{d}} = \max_{\mathbf{d} \in R} [c(\mathbf{d})] \quad (6.2)$$

There are two widely adopted choices for  $\psi(u, v)$ , namely *sum of squared differences (SSD)* where

$$\psi(u, v) = -(u - v)^2 \quad (6.3)$$

and *cross correlation* where

$$\psi(u, v) = uv \quad (6.4)$$

Due to varying lighting conditions, the similarity condition for correlation is not optimal, and we rather use *normalized cross correlation*, which is defined as follows:

$$\psi(u, v) = \frac{(u - \bar{u})(v - \bar{v})}{N_u N_v} \quad (6.5)$$

where  $W$  is our search window and

$$\begin{aligned} \bar{u} &= \frac{1}{W} \sum_W u \\ \bar{v} &= \frac{1}{W} \sum_W v \\ N_u &= \frac{1}{W} \sqrt{\sum_W u^2} \\ N_v &= \frac{1}{W} \sqrt{\sum_W v^2} \end{aligned} \quad (6.6)$$

This similarity measure is invariant to lighting intensities, and gives much better results.

The search window denoted by  $W$  is a free parameter, which should be adjusted to best suit the observed scene. In highly textured scenes a small window should suffice, but indoor scenes generally require a larger window due to their large areas with little or no features.

Unfortunately there is still no single method or set of parameters giving optimal results. Different scenes require different methods and/or parameters. We have chosen to implement the above-mentioned *normalized cross correlation* due to its simplicity, it returns dense disparity maps, and the possibility of implementing it highly optimized.

### 6.3 Constraints

There are several factors influencing the effectiveness of the correspondence problem. We will name just a few:

- *Varying lighting conditions* : Even though normalized cross correlation attempts to correct for this, varying lighting can create dark shadows and bright reflections which adversely affect the correspondence problem.
- *Occlusions* : Feature points in one image may not have a valid match in the other. This is due to not all objects being visible in both images. The matching algorithm needs to be aware of this possibility and detect it when it occurs.
- *Spurious matches* : Incorrect matches are sometimes found due to noise, or several of the other factors mentioned here. Every effort should be made to detect this.
- *Incorrect search windows* : The search window,  $W$ , needs to be placed in the right position and big enough to ensure the corresponding feature lies inside. However, as the window size increases, so does the chance of a spurious match. The optimal size can be determined from the epipolar geometry.
- *Featureless Areas* : Consider the following scenario; a sub-image of a white wall is being correlated to another stretch of white wall. The correlation values returned throughout is very close to unity, indicating a good match throughout the search area. The matching algorithm should detect this situation, and rather report no match than a false match.

Due to all the problems involved in correspondence, it is sometimes referred to as an *ill-posed* problem. However, there are a number of *constraints* we can use to simplify the problem, or reduce the search space. This enables us to obtain practically useful disparity maps.

We will now discuss a few constraints and their implementations.

### 6.3.1 Epipolar Constraint

After selecting a feature in the first image, we perform a two-dimensional search in the second image to obtain a match. It is possible to, through knowledge of the epipolar geometry, reduce this search space to one dimension. Consider the following case. Two intrinsically identical cameras observe the same scene. They are separated only on the  $X$  axis, and their alignment is identical. This entails that the inter-camera rotation matrix  $R$  is identity, and the translation vector  $\mathbf{T}$  has only a horizontal element. Now if we select a feature in the left image, we need only look along the same scanline in the right image to obtain a match. This is due to the cameras being exactly parallel. This is the essence of epipolar rectification.

If we refer back to chapter 4 to the definition of epipolar geometry, we see that the fundamental matrix has exactly the information we need. The practical importance of

the epipolar geometry is that for a feature in the left image, we can constrain the search for its corresponding feature in the right image to the *epipolar line*. If our feature in the left image is denoted  $\mathbf{p}_l$  and the fundamental matrix by  $F$ , then:

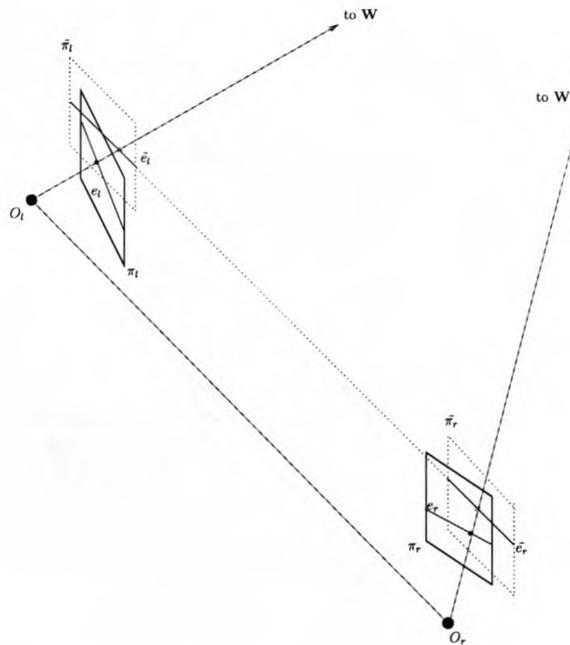
$$\mathbf{u}_r = F\mathbf{p}_l \quad (6.7)$$

where  $\mathbf{u}_r$  is the projective epipolar line, constraining our search space to one dimension. Using this constraint results in a huge speed increase, as well as increased accuracy. The problem is that the epipolar lines along which we search tend to be across scan-lines, and this complicates and produces unnecessary bottlenecks in the processing. We will now explain how to rectify this, by forcing corresponding lines to be horizontal.

### Planar Epipolar Rectification

Planar epipolar rectification is a well known topic in the stereo vision community. Most matching algorithms assume that images are already epipolarly rectified, which implies that corresponding features lie on the same scanline, and thus an effective and fast method is needed for rectification, since it has to be performed before matching can be done.

Given a pair of stereo images, *rectification* determines a transformation of each image plane such that pairs of conjugate epipolar lines become collinear and parallel to one of the image axes, usually the horizontal one. The important advantage of rectification is that computing stereo correspondences is made simpler and more accurate, since only a one-dimensional search needs to be done.



**Figure 6.2:** *Planar Epipolar Rectification*

Figure 6.2 illustrates the situation. Two cameras, their centers denoted by  $\mathbf{O}_l$  and  $\mathbf{O}_r$ , are observing a point  $\mathbf{P}$ . The original retinal planes drawn with solid lines are denoted by  $\pi_l$  and  $\pi_r$ , while the new rectified planes drawn with dotted lines are labelled  $\tilde{\pi}_l$  and  $\tilde{\pi}_r$ . The epipolar lines corresponding to the projections  $\mathbf{p}_l$  and  $\mathbf{p}_r$  of  $\mathbf{P}$ , are denoted by  $\mathbf{e}_l$  and  $\mathbf{e}_r$ . It can be seen that these epipolar lines are *not* horizontal. Epipolar rectification is the process of forcing these lines to be horizontal. This is achieved by generating two *virtual cameras* in space, which in effect implies the rotation of the two retinal planes. These rotated retinal planes are shown in dotted lines, and it can be seen that the new epipolar lines,  $\tilde{\mathbf{e}}_l$  and  $\tilde{\mathbf{e}}_r$ , are horizontal.

The problem of epipolar rectification is to determine the transform to achieve this. Most of the known methods fall into this category of planar rectification, where, in essence, the original cameras are rotated to acquire new images. The main difference between the algorithms lie in how they determine some of the undefined parameters. The distinction between necessary and arbitrary constraints are unclear. We will now detail a planar epipolar rectification algorithm proposed by Fusiello et al [13], which is well defined, simple and compact.

First some definitions to simplify the algorithm:

The perspective projection matrix of a camera,  $P$ , can be written as

$$P = A[R|\mathbf{T}] \quad (6.8)$$

where  $A$  is the intrinsic matrix defined in (3.12) and  $[R|\mathbf{T}]$  is the rotation matrix and translation vector respectively.

We write the projection matrix  $P$  as,

$$P = \left[ \begin{array}{c|c} \mathbf{q}_1^T & q_{14} \\ \mathbf{q}_2^T & q_{24} \\ \mathbf{q}_3^T & q_{34} \end{array} \right] = [Q|\bar{\mathbf{q}}] \quad (6.9)$$

The *focal plane* is the plane parallel to the retinal plane that contains the optical center  $C$ . The coordinates,  $\mathbf{c}$  of  $C$  are given by [13]:

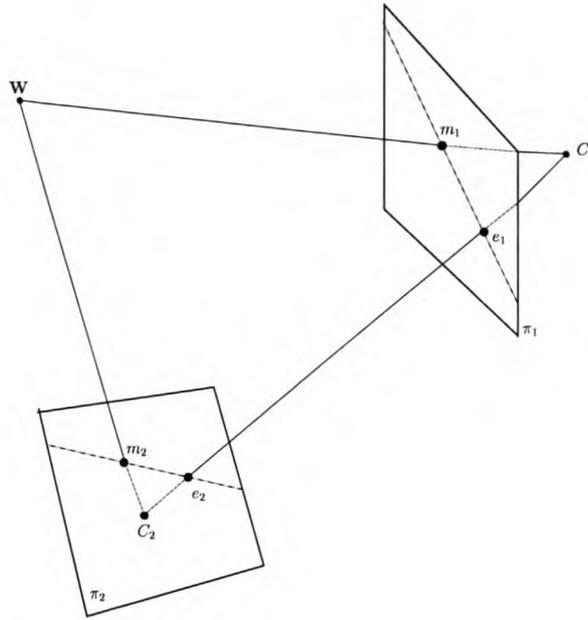
$$\mathbf{c} = -\mathbf{Q}^{-1}\bar{\mathbf{q}} \quad (6.10)$$

Therefore  $\mathbf{P}$  can be written as:

$$\mathbf{P} = [Q | -Q\mathbf{c}] \quad (6.11)$$

The *optical ray* associated to an image point  $\mathbf{m}$  is the line  $\mathbf{m} \rightarrow \mathbf{C}$ . The equation of this ray can be written in parametric form

$$\mathbf{w} = \mathbf{c} + \lambda\mathbf{Q}^{-1}\mathbf{m} \quad (6.12)$$



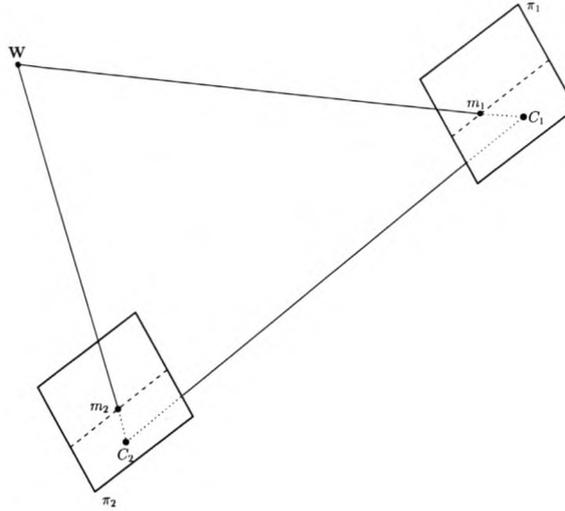
**Figure 6.3:** *Unrectified image planes*

with  $\mathbf{w}$ ,  $\mathbf{c}$  and  $\mathbf{m}$  given in homogeneous coordinates.

Now refer to figure 6.3 with two cameras with  $\mathbf{C}_1$  and  $\mathbf{C}_2$  their respective optical centers. A 3D point,  $\mathbf{W}$ , is projected onto both image planes, to points  $\mathbf{m}_1$  and  $\mathbf{m}_2$ , which constitute a conjugate pair. Given the point  $\mathbf{m}_1$  in the left image, it's conjugate in the right image is constrained to lie on the epipolar line of  $\mathbf{m}_1$ . Since  $\mathbf{m}_1$  may be the projection of any point on it's optical ray, the epipolar line is the projection through  $\mathbf{C}_2$  of the optical ray of  $\mathbf{m}_1$ . All the epipolar lines in one image plane pass through a common point ( $\mathbf{e}_1$  and  $\mathbf{e}_2$  respectively), called the epipoles, which are the projections of the conjugate optical centers.

When  $\mathbf{C}_1$  is in the focal plane of the right camera, the right epipole is at infinity, and the epipolar lines form a bundle of parallel lines in the right image. A very special case is when both epipoles are at infinity, when the line  $\mathbf{C}_1\mathbf{C}_2$  is contained in both focal planes. In this case the epipolar lines form a bundle of parallel lines in both images, which is the objective of *epipolar rectification*. Any pair of images can be transformed so that epipolar lines are parallel and horizontal in each image.

The algorithm now proceeds as follows; We assume that the stereo rig is calibrated, that is the projection matrices  $P_l$  and  $P_r$  are known. Since the cameras are calibrated we also know the extrinsic and intrinsic parameters for both cameras. We now wish to determine two new projection matrices,  $\tilde{P}_l$  and  $\tilde{P}_r$ , obtained by rotating the old projection matrices around their optical centers until the focal planes become coplanar, thereby containing the baseline. To ensure horizontal epipolar lines, the baseline must be parallel to the new  $X$  axis of both cameras. The requirement that conjugate points must lie on



**Figure 6.4:** *Rectified image planes*

the same vertical coordinate is met by forcing both cameras to have the same intrinsic parameters. Figure 6.4 shows the geometry we want to obtain.

Let us now write the new projection matrices in terms of their intrinsic and extrinsic parameters (6.8) and (6.11)

$$\begin{aligned}\bar{P}_l &= A[R] - Rc_1 \\ \bar{P}_r &= A[R] - Rc_2\end{aligned}\tag{6.13}$$

The intrinsic matrix for both new projection matrices is the same, and can be chosen arbitrarily. We adjust this matrix to ensure we obtain the required size images in the right positions, by manipulating the image center and focal length. The optical centers,  $\mathbf{c}_1$  and  $\mathbf{c}_2$ , which stay constant, are given by equation (6.10). The matrix,  $R$ , which determines the camera pose, is the same for both projection matrices, and will be specified by its row vectors.

$$R = \begin{bmatrix} \mathbf{r}_1^T \\ \mathbf{r}_2^T \\ \mathbf{r}_3^T \end{bmatrix}\tag{6.14}$$

determines the  $X$ ,  $Y$  and  $Z$  axis respectively. According to the previously stated requirements, we choose [13]:

- The new  $X$  axis,  $\mathbf{r}_1$ , parallel to the baseline,
- The new  $Y$  axis,  $\mathbf{r}_2$ , orthogonal to  $X$  (mandatory) and orthogonal to  $\mathbf{k}$ ,

- Then new  $Z$  axis,  $\mathbf{r}_3$ , orthogonal to  $X$  and  $Y$ ,

where  $\mathbf{k}$  is an arbitrary unit vector that fixes the position of the new  $Y$  axis in the plane orthogonal to  $X$ . We take it equal to the  $Z$  unit vector of the left camera, thereby constraining the new  $Y$  axis to be orthogonal to both the new  $X$  and old left  $Z$  axes. This discussion clearly demonstrates that there are many possible choices for how the axes are chosen, and thus many subtly different rectification algorithms. The row vectors of  $R$  are now defined as,

$$\begin{aligned}\mathbf{r}_1 &= \frac{(\mathbf{c}_1 - \mathbf{c}_2)}{\|\mathbf{c}_1 - \mathbf{c}_2\|} \\ \mathbf{r}_2 &= \mathbf{k} \times \mathbf{r}_1 \\ \mathbf{r}_3 &= \mathbf{r}_1 \times \mathbf{r}_2\end{aligned}\tag{6.15}$$

Now we have defined our new projection matrices, and need to derive the necessary image transformations.

In order to rectify, for example, the left image we need to compute the transformation mapping the image plane of  $P_l$  onto the image plane of  $\bar{P}_l$ . We will see that the desired transformation is the collinearity given by the 3x3 matrix  $B_l$ , (see [13])

$$B_l = \bar{Q}_l Q_l^{-1}\tag{6.16}$$

with  $\bar{Q}_l$  extracted from  $\bar{P}_l$  and  $Q_l$  from  $P_l$ . The same applies to the right image.

Now for any 3D point  $\mathbf{W}$  we can write

$$\begin{aligned}\mathbf{w}_1 &= P_l \mathbf{W} \\ \bar{\mathbf{w}}_1 &= \bar{P}_l \mathbf{W}\end{aligned}\tag{6.17}$$

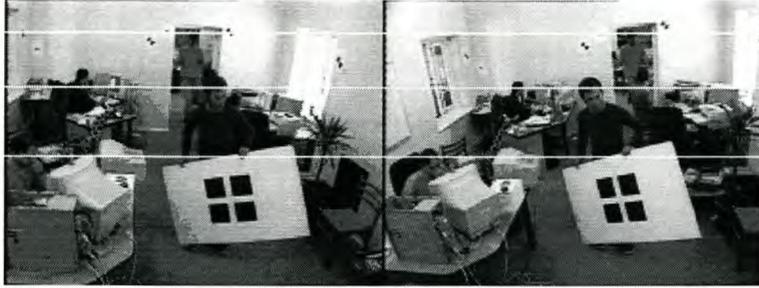
where  $\mathbf{w}_1$  and  $\bar{\mathbf{w}}_1$  are the old and new projections of  $\mathbf{W}$  respectively. Since the optical center is constant it follows (6.12) that the equations of the optical rays are given by

$$\begin{aligned}\mathbf{w} &= \mathbf{c}_1 + \lambda \mathbf{Q}_1^{-1} \mathbf{m}_1 \\ \mathbf{w} &= \mathbf{c}_1 + \bar{\lambda} \bar{\mathbf{Q}}_1^{-1} \bar{\mathbf{m}}_1\end{aligned}\tag{6.18}$$

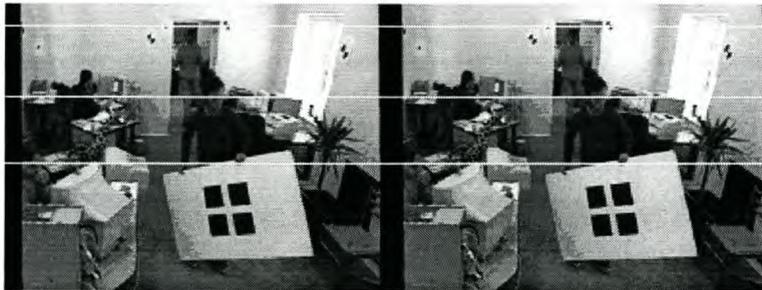
and hence

$$\bar{\mathbf{m}}_1 = \lambda \bar{\mathbf{Q}}_1 \mathbf{Q}_1^{-1} \mathbf{m}_1\tag{6.19}$$

where  $\lambda$  is an arbitrary scale factor. This shows that we can apply the transformation  $\mathbf{B}_l$  directly to the left image coordinates to obtain the rectified image. These new coordinates are generally non-integer and the grey levels are thus determined by bilinear interpolation.



**Figure 6.5:** *Epipolar Unrectified images*



**Figure 6.6:** *Epipolar Rectified Images*

Using matches obtained from our rectified images we can then perform triangulation directly using the new projection matrices,  $\bar{P}_l$  and  $\bar{P}_r$ .

A practical example is shown in figures 6.5 and 6.6. Figure 6.5 shows the left and right images taken from our stereo cameras. We added three horizontal white lines to highlight the vertical difference between corresponding physical features. It can be seen that a physical feature in the left image which lies on the line does not lie on it in the right image. They thus have different  $y$  coordinates. This can be most clearly seen on the black dot against the wall in the top left corner of the image.

Figure 6.6 shows the same two images which have now been rectified using the method described above. We plotted three horizontal lines again to aid the user in comparisons. The black dot against the window in the top left corner now lies on exactly the same scanline in both images. This is consistent with all the physical features in the image. The search for corresponding features can now be limited to a horizontal search, which was the goal we wanted to achieve. Note that the rectified images are distorted versions of the originals. We can adjust the size and aspect ratio of the images by modifying the arbitrary intrinsic matrix used in (6.13).

### **Polar Epipolar Rectification**

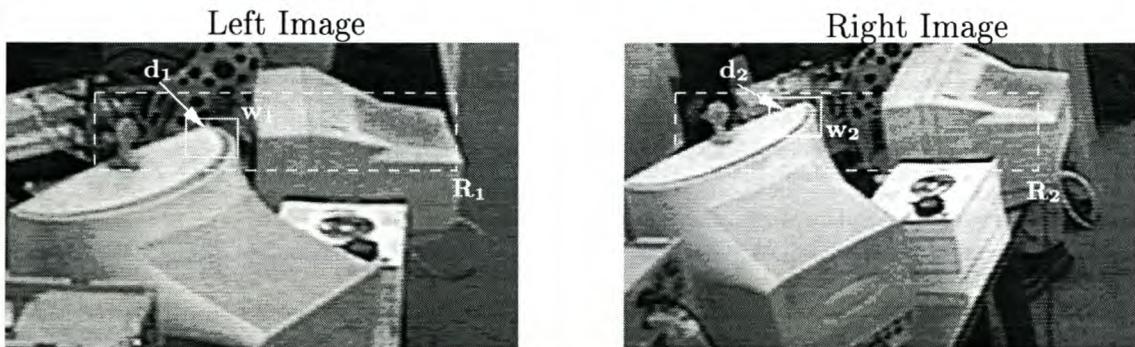
For traditional stereo applications the limitations of planar rectification algorithms are not so important. The main component of camera displacement is parallel to the images for

classical stereo setups. However, new advances in structure and motion make it possible to retrieve 3D models of scenes acquired with hand-held cameras. In these cases it is quite possible to have forward motion resulting in the epipoles lying inside the images. In these cases, an alternative to the standard planar rectification is required. This is achieved via *polar rectification*, an algorithm presented by Pollefeys et al [30].

The key to this rectification method consists of reparametrizing the image with polar coordinates centered around the epipoles. Using this method the image can be *unrolled* around the epipole, solving the problem of having an epipole in the image. This method was not implemented by us, due to the standard stereo setup we used, and the reader is referred to [30] for more detail on the algorithm.

### 6.3.2 Left-Right Constraint

The left-right constraint is a very simple procedure to implement, but it gives significant improvements on validity. It can be defined as, a feature in the left image has only one match in the right image, and *this* feature matches only to the same feature in the left image.



**Figure 6.7:** *Left-Right Constraint*

We implement this procedure as shown in figure 6.7. The image region on the left,  $w_1$  is matched throughout the search region  $R_2$  on the right. The similarity measure peaks at position  $d_2$ , indicating this is the conjugate feature. The uniqueness constraint now says that the feature at  $d_2$ , namely  $w_2$ , if matched onto the search area  $R_1$  on the left image, must correspond to  $d_1$ , which is the original window. This is also known as the *left-right consistency* constraint. It entails that the matching algorithm for an image has to be performed twice, left-to-right and right-to-left, but we found that this was worth the extra calculations.

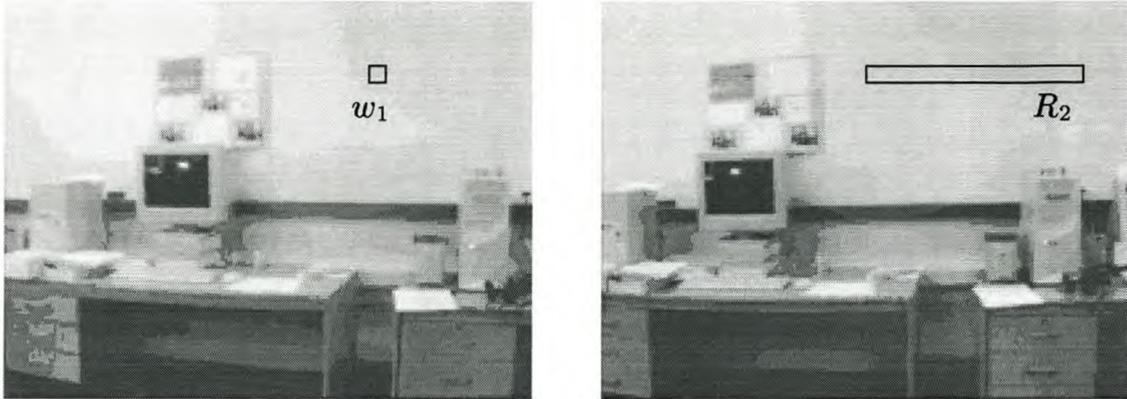


Figure 6.8: *Validity Measure*

### 6.3.3 Continuity

This is a difficult constraint to enforce, since it has two conflicting ideals. It is assumed that observed objects have piecewise continuous surfaces, implying that the disparity map produced should be relatively smooth, with no sharp changes in depth. However, many scenes do produce what seems like discontinuities in images, especially man made scenes with sharp edges. The matching algorithm has to enforce the continuity constraint on smooth areas, but disregard it at discontinuities. Some work has been done on this, notably [20].

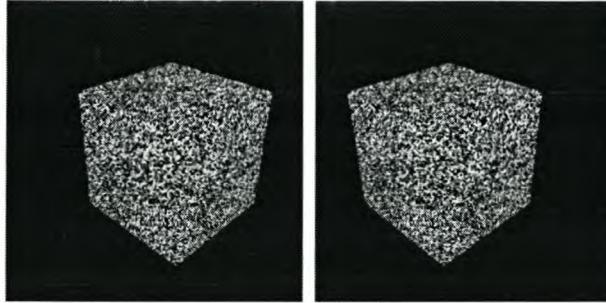
### 6.3.4 Validity Measure

Figure 6.8 demonstrates the scenario. If the image section  $w_1$  is matched across the search area  $R_2$ , it produces correlation values which stay very close to 1.0. This is due to the fact that both the feature window,  $w_1$ , and the search area  $R_2$  are very similar. The matching algorithm will now return the disparity with the highest similarity measure as the correct match, where in fact it achieved very similar matches throughout the search area.

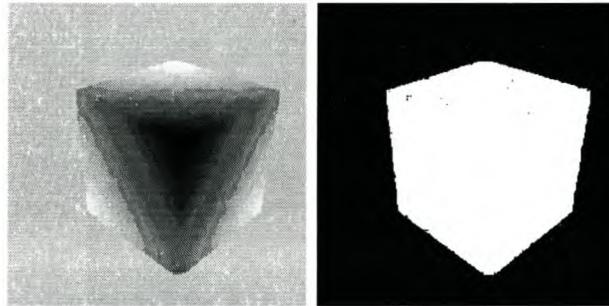
We try to get an indication of this scenario by taking the correlation values,  $C_n$ , where  $n \in [d_1, d_2]$  as statistical samples, and calculating the mean and variance of these samples. We now get an accurate indication of whether the correlation values stayed very constant over the search area, which indicates a situation like the above. This enables us to rather disregard the disparity for this feature than accepting an incorrect one.

## 6.4 Speed

Stereo vision applications are often required to work real-time, such as in tasks like remote navigation and on-line tracking of objects. The definition of real-time depends greatly



**Figure 6.9:** *Synthetic Cube for Disparity Calculation*



**Figure 6.10:** *Disparity and Validity for Synthetic Cube*

on the task involved. For example planetary rover could have a transmission time lag in the order of minutes and a time delay of 10 seconds per frame would be quite acceptable. However, in an application like collision detection for indoor robots, a much lower delay would be required.

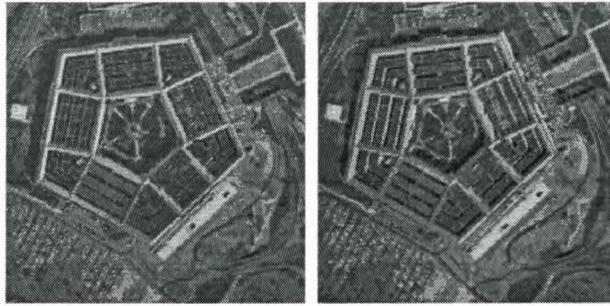
Matching algorithms are inherently slow due to the massive amount of calculations which need to be performed, as well as large search spaces. The first solution to speeding them up is to reduce the search space, which we achieve by using the epipolar constraint. Secondly we looked at the algorithm itself to enhance the performance.

The algorithm itself could be significantly sped up by storing calculated data which would be used again in later calculations, instead of recalculating. Significant gains were also achieved by using the MMX instructions of the Intel Pentium III processors, which are Single-Instruction-Multiple-Data commands. We obtain average speeds of approximately 1.7 seconds per frame on a Pentium III 750 MHz. This includes calculating disparity left-to-right and right-to-left, as well as thresholding correlation scores and interpolating disparity results.

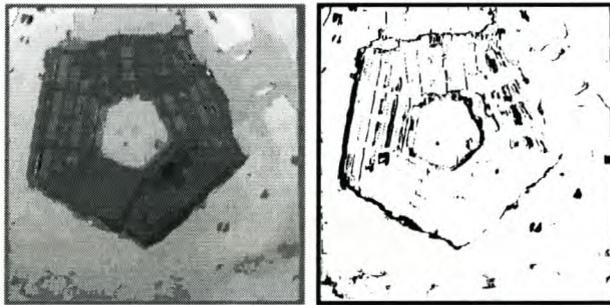
## 6.5 Results

We will firstly demonstrate results obtained with synthetic data. Figure 6.9 shows the left and right views of a cube generated by a ray tracing program. These images are supplied

by Bill Hoff of the University of Illinois. The images are already epipolarly rectified, and the disparity ranges from 15 to 50. Our algorithm produced the two pictures of figure 6.10. The left image shows the disparity found, with darker pixels being closer to the viewer than light pixels. The image on the right is a boolean map showing where valid correspondences were found, with white being valid and black invalid. Note that the entire background was found invalid thanks to the *validity measure*.



**Figure 6.11:** *Aerial Pentagon view for Disparity Calculation*



**Figure 6.12:** *Disparity and Validity for Pentagon*

The next example is an aerial view of the Pentagon, supplied by Carnegie-Mellon University [1], which is often used by researchers to demonstrate disparity algorithms. Figure 6.11 shows the left and right views of the Pentagon. Figure 6.12 once again shows the disparity on the left and the validity on the right. Darker pixels on the disparity are closer to the viewer, and lighter pixels further away. The validity map shows correct disparities with white pixels, and incorrect with black.

It can be seen that the disparity algorithm works effectively and generates dense disparity maps which can be used to determine the depth using triangulation.

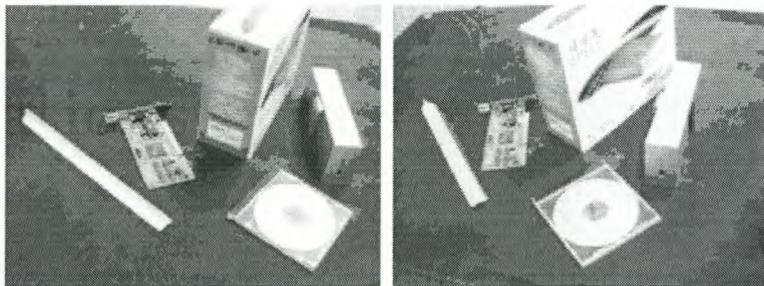
# Chapter 7

## Applications

This chapter serves as an example to demonstrate how the stereo camera system could function on real world data. Since there are endless applications for such a system, we picked two to demonstrate some of the features and possible uses.

### 7.1 Computer Vision Metrology

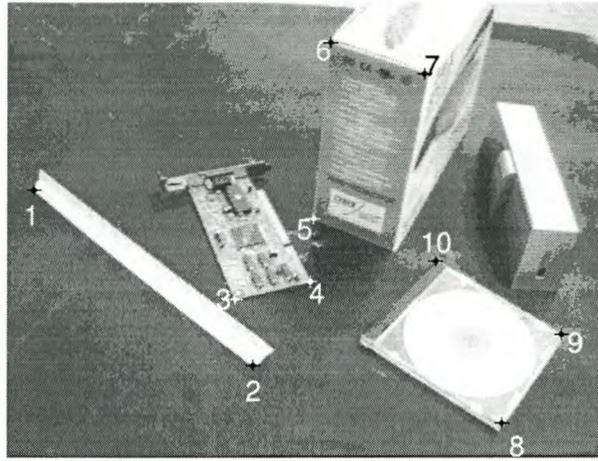
Using a fully calibrated stereo rig enables us to measure exact physical distances from the images using triangulation. A system like this could be used in an industrial manufacturing plant to do automatic product inspection and a multitude of other tasks.



**Figure 7.1:** *Stereo Images for Measurements*

Figure 7.1 shows the left and right images from a stereo rig. This rig was fully calibrated after which we placed the objects in the field of view of the camera. We now proceeded to determine the image coordinates of ten features on both images. This was done by hand, and is therefore not very accurate. In figure 7.2 we show these features for the left image with their corresponding index numbers. The image coordinates of these features for both images are shown in table 7.1. Note that to obtain a high level of accuracy these should be extracted to sub-pixel accuracy.

By using the image matches given in table 7.1 we could now perform triangulation to measure some physical distances. These we compared to what we measure using a



**Figure 7.2:** *Index of Measurement Points*

<i>Feature</i>	<i>Left Coordinate</i>	<i>Right Coordinate</i>
1	(34, 238)	(59, 215)
2	(316, 459)	(158, 410)
3	(292, 375)	(181, 338)
4	(387, 353)	(268, 326)
5	(391, 273)	(316, 258)
6	(416, 50)	(252, 46)
7	(541, 50)	(334, 82)
8	(630, 528)	(387, 523)
9	(708, 421)	(540, 428)
10	(548, 325)	(426, 315)

**Table 7.1:** *Feature Coordinates*

standard ruler, which serves as ground truth. Note that due to human error this would only be accurate to within 1 millimetre. In table 7.2 we show the results of this experiment. The first two columns specify between which two features we measured. The third column is what we measured using a ruler, and the final column the result of our triangulation routine. It can be seen that a reasonable accuracy can easily be obtained. The level of accuracy can be increased by using a sub-pixel corner extractor, higher resolution images and precision manufactured calibration objects.

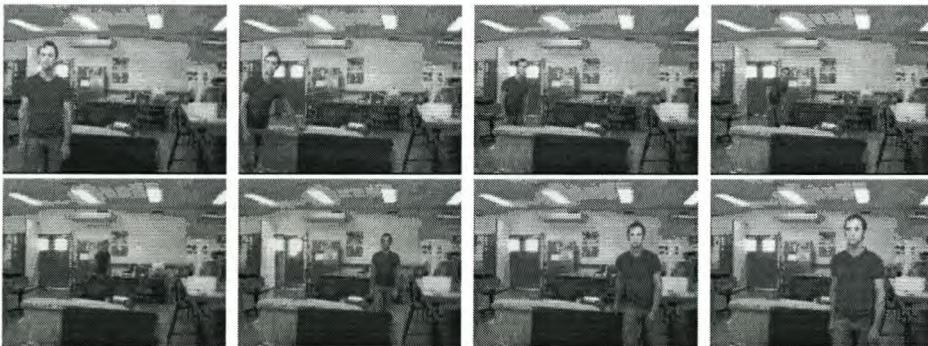
The calibration object used for this experiment consisted of two planes with black squares which were printed on a standard laser printer, which introduces some positioning errors.

<i>From</i>	<i>To</i>	<i>Physical</i>	<i>Triangulation</i>
1	2	320 mm	319.385 mm
3	4	72 mm	71.80 mm
5	6	205 mm	203.78 mm
6	7	87 mm	85.79 mm
8	9	120 mm	118.83 mm
9	10	141 mm	139.12 mm

**Table 7.2:** *Comparison of Physical Distances and Triangulation*

## 7.2 3D Tracking

Another application which is becoming more popular is for security applications. The idea is to track people in three dimensions walking through a building. This would entail that there is always at minimum two cameras observing the target. This would allow us to triangulate and thus obtain a three-dimensional route that the target followed.

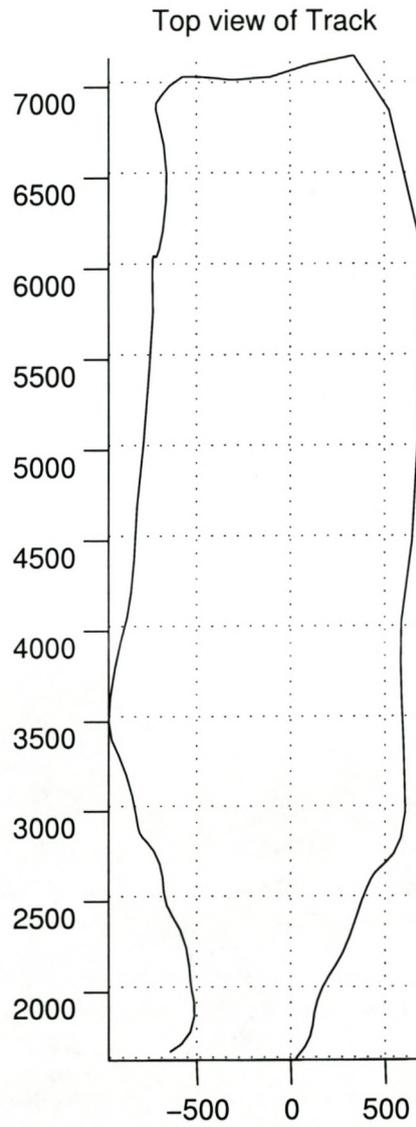


**Figure 7.3:** *Selected images of left camera person tracking*

We performed this experiment on a small scale using two cameras. Figure 7.3 shows a number of images from the tracking sequence used. We grabbed 100 images from each camera, and tracked the position of the subject's using a combination of an in-house developed face-tracking algorithm and manual tracking. These coordinates were used to triangulate the 3D position of the person in relation to the left camera.

Figure 7.4 shows the top view of the 3D track the person followed. This track was obtained from the triangulation of the coordinates of the subject's head and then smoothed to reduce the effects of measurement noise. The axis values are given in millimetres, which corresponded very well with our own measurements.

This example simply serves to demonstrate the versatility of the stereo system. It can be used in a variety of different applications and produces accurate results in real world situations.



**Figure 7.4:** *Top View of track*

## Chapter 8

### Conclusion

The goal of this thesis was to develop and implement an automatic and accurate stereo camera calibration method. This was achieved through a variety of steps. The problem of automatically locating the calibration object and extracting the features needed for calibration has traditionally required human intervention. We developed an automatic algorithm to locate the calibration object and extract the features to sub-pixel accuracy. This was shown to work accurately even in very cluttered environments, thus enabling us to streamline the calibration procedure in practical situations.

The process of determining the intrinsic parameters of cameras has been extensively studied and we opted to use a well known calibration method. As the results in chapter 3 show we obtain parameters very similar to other industry standard methods which is to be expected since we utilise the same algorithms. The slightly better performance of our implementation can be attributed to a more efficient nonlinear optimizer.

Our biggest contribution is the stereo camera calibration algorithm proposed in chapter 4. This method combined multiple images into one data set and was shown to perform very accurately. In our experiments it performs much better in the presence of noise than currently available methods. It is also very flexible in that it can use any form of calibration object and using more calibration images increases the accuracy and robustness.

The algorithms for developing a fully fledged stereo system were also described and shown to work accurately and efficiently. We detailed a planar epipolar rectification method which eases the task of finding correspondences. The entire correspondence problem was also discussed in detail and we presented some constraints which increase the accuracy of the correspondence calculations and produce a map showing where valid correspondences were obtained.

We conclude that the main goals of this thesis were achieved, namely developing an accurate, automatic and easy to use stereo calibration method.

# Bibliography

- [1] “Calibrated Imaging Laboratory, Carnegie-Mellon Univeristy.”  
<http://www-2.cs.cmu.edu/afs/cs/project/cil/www/>. January 2002.
- [2] “Intel Open Source Computer Vision Library.”  
<http://www.intel.com/research/mrl/research/opencv/>. January 2002.
- [3] “JAI Camera Solutions.” [www.jai-camera-solutions.com](http://www.jai-camera-solutions.com). January 2002.
- [4] “Jean Yves Bouguet Camera Calibration Toolbox.”  
<http://www.vision.caltech.edu>. January 2002.
- [5] “Linux Operating System.” [www.linux.com](http://www.linux.com). January 2002.
- [6] “Reg Willson’s Homepage.”  
<http://telerobotics.jpl.nasa.gov/people/rwillson/>. January 2002.
- [7] “Zoran Corporation.” [www.zoran.com](http://www.zoran.com). January 2002.
- [8] BARTOLI, A. and STURM, P., “Three New Algorithms for Projective Bundle Adjustment with Minimum Parameters.” Tech. Rep. RR-4236, INRIA, Sophia-Antipolis, 2001.
- [9] BOUQUET, J.-Y. and PERONA, P., “3D Photography on your Desk.” *Proc. 6th International Conference on Computer Vision*, 1998, pp. 43–50.
- [10] CHEN, X., DAVIS, J., and SLUSALLEK, P., “Wide Area Camera Calibration Using Virtual Calibration Objects.” *Proceedings of the Computer Vision and Pattern Recognition*, 2000.
- [11] FAUGERAS, O., *Three-Dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, Cambridge, Massachusetts, 1993.
- [12] FAUGERAS, O. and TOSCANI, G., “The calibration problem for stereo.” in *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 15–20, 1986.

- [13] FUSIELLO, A., TRUCCO, E., and VERRI, A., "A compact algorithm for rectification of stereo pairs." *Machine Vision and Applications*, 2000, Vol. 12, pp. 16–22.
- [14] H. A. MARTINS, J. R. B. and KELLEY, R. B., "Camera Models Based on Data from Two Calibration Planes." *Computer Graphics and Image Processing*, 1981, Vol. 17, pp. 173–180.
- [15] HARRIS, C. and STEPHENS, M., "A combined corner and edge detector." *Proc. 4th Alvey Vision Conference*, 1988, pp. 147–151.
- [16] HARTLEY, R. and STURM, P., "Triangulation." *Computer Vision and Image Understanding*, 1997, Vol. 68, pp. 146–157.
- [17] HARTLEY, R. I., "An Algorithm for Self Calibration from Several Views." in *International Computer Vision and Pattern Recognition Conference*, pp. 908–912, 1994.
- [18] HERSHBERGER, J. and SNOEYINK, J., "Speeding Up the Douglas-Peucker Line-Simplification Algorithm." *Proc. of the 5th International Symposium on Spatial Data Handling*, 1992, Vol. 1, pp. 134–143.
- [19] ISAGUIRRE, A., PU, P., and SUMMERS, J., "A new Development in Camera Calibration: Calibrating a Pair of Mobile Vectors." in *Proc. Int. Conference on Robotics and Automation*, 1985.
- [20] ISHIKAWA, H. and GEIGER, D., "Occlusions, discontinuities, and epipolar lines in stereo." in *Proc. European Conference on Computer Vision*, pp. 232–248, 1998.
- [21] LENZ, R. and TSAI, R., "Techniques for calibration of scale factor and image center for high accuracy 3d vision metrology." in *Proc. IEEE Int. Conference on Robotics and Automation*, 1987.
- [22] LI, N., "Retrieving Camera Parameters from Real Video Images." Master's thesis, The University of British Columbia, 1988.
- [23] LONGUET-HIGGINS, H., "A computer algorithm for reconstructing a scene from two projections." *Nature*, 1981, Vol. 293, pp. 133–135.
- [24] LUONG, Q. and FAUGERAS, O., "The fundamental matrix: theory, algorithms, and stability analysis." in *International Journal of Computer Vision*, vol. 17, pp. 43–75, 1996.

- [25] LUONG, Q. T. and FAUGERAS, O., "Self-calibration of a stereo rig from unknown camera motions and point correspondences." Tech. Rep. RR-2014, INRIA, Sophia-Antipolis.
- [26] LUONG, Q. and FAUGERAS, O., "Self-calibration of a moving camera from point correspondences and fundamental matrices." *International Journal of Computer Vision*, 1997, Vol. 22, pp. 261–289.
- [27] MORAVEC, H., "Towards automatic visual obstacle avoidance." *Proc. International Joint Conference on Artificial Intelligence*, 1977, p. 584.
- [28] NIBLACK, W., *An Introduction to Digital Image Processing*. New York: Prentice-Hall International, 1986.
- [29] POLLEFEYS, M., "Tutorial on 3D Modelling from Images." tech. rep., Katholieke Universiteit Leuven, 2000.
- [30] POLLEFEYS, M., KOCH, R., and GOOL, L. V., "A simple and efficient rectification method for general motion." in *Proc. International Conference on Computer Vision*, 1999.
- [31] PROESMANS, M., VAN GOOL, L., and DEFOORT, F., "Reading between the lines - a method for extracting dynamic 3D with texture." *Proc. 6th International Conference on Computer Vision*, 1998, pp. 1081–1086.
- [32] TRIER, O. and JAIN, A., "Goal-Directed Evaluation of Binarization Methods." *IEEE Transactions on Pattern Recognition and Machine Intelligence*, December 1995, pp. 1191–1201.
- [33] TRUCCO, E. and VERRI, A., *Introductory techniques for 3-D computer vision*. Prentice Hall, 1998.
- [34] TSAI, R., "Synopsis of recent progress on camera calibration for 3D machine vision." in *The Robotics Review*, pp. 147–159, MIT Press, 1989.
- [35] TSAI, R., "A Versatile Camera Calibration Technique for High-Accuracy 3-D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses." in *Radiometry – (Physics-Based Vision)* (WOLFF, L., SHAFER, S., and HEALEY, G. (Eds)), Jones and Bartlett, 1992.
- [36] WEI, G. and MA, S., "Implicit and explicit camera calibration: Theory and experiments." *IEEE transactions on Pattern Analysis and Machine Intelligence*, May 1994, Vol. 16, pp. 469–480.

- [37] WILLSON, G., REG. and SHAFER, A., STEVEN, "What is the Center of the Image?." Tech. Rep. CMU-CS-93-122, Carnegie-Mellon University, April 1993.
- [38] ZELLER, C. and FAUGERAS, O., "Camera Self-Calibration from Video Sequences: the Kruppa Equations Revisited." Tech. Rep. RR-2793, INRIA, Sophia-Antipolis.
- [39] ZHANG, Z., "Determining the epipolar geometry and its uncertainty: A review." Tech. Rep. RR-2927, INRIA, Sophia-Antipolis, 1996.
- [40] ZHANG, Z., DERICHE, R., FAUGERAS, O., and LUONG, Q.-T., "A Robust Technique for Matching two Uncalibrated Images Through the Recovery of the Unknown Epipolar Geometry." *Artificial Intelligence*, 1995, Vol. 78, pp. 87–119.
- [41] ZHANG, Z., "A Flexible New Technique for Camera Calibration." *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000, Vol. 22, pp. 1330–1334.