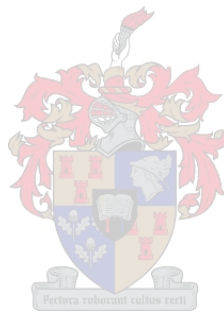


Rapid 3D Measurement Using Digital Video Cameras

by

Willem Johannes Van der Merwe



March 2008

Rapid 3D Measurement Using Digital Video Cameras

by

Willem Johannes Van der Merwe

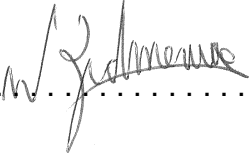
*Thesis presented in partial fulfillment of the requirements for
the degree of Master of Science in Mechatronic Engineering at
the University of Stellenbosch*

Supervisor: Dr. K. Schreve

March 2008

Declaration

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

Signature: 

W. J. Van der Merwe

Date: 05 • 03 • 2008

Copyright © 2008 University of Stellenbosch
All rights reserved.

Abstract

A rapid measurement system is implemented using two digital video cameras, presenting a faster and less expensive solution to certain metrology problems.

The cameras are calibrated from one stereo image-pair of a 3D calibration grid that allows an immediate assessment of the achievable metric accuracy of the system. Three different methods, using either laser tracking or structured light patterns, were developed and employed to solve the coordinate extraction and correspondence matching problems. Different image processing techniques were used to speed up the entire measurement process. All software development was accomplished using only freely distributed software packages.

The system achieves calibration in less than a minute and accumulates point correspondences at 12 frames per second. Accuracies of greater than 0.4 mm are achieved for a 235 x 190 x 95 mm measurement volume using a single pair of images with 640 x 480 pixel resolution each.

Uittreksel

Met die gebruik van twee digitale videokameras word 'n spoedige meetsisteem geïmplementeer om sodoende 'n vinniger en meer bekostigbare oplossing te bied vir sekere metrologie probleme.

Die kameras word gekalibreer deur een beeldpaar van 'n 3D kalibrasieveld te gebruik wat oombliklike assessering van die behaalbare akkuraatheid van die sisteem moontlik maak. Drie metodes is ontwikkel en geïmplementeer om die probleme van koördinaatontginning en puntooreenstemming op te los. Die laasgenoemde metodes maak gebruik van of ligpatrone of laser. Verskillende beeldverwerkingsmetodes word gebruik om die meetproses te verspoedig. Alle sagtewareontwikkeling is met vrylik beskikbare sagteware pakette gedoen.

Die sisteem kan gekalibreer word in minder as 'n minuut en versamel puntooreenstemmings teen 'n tempo van 12 per sekonde. Akkuraatheid van beter as 0.4 mm word behaal vir 'n meetvolume van 235 x 190 x 95 mm deur gebruik te maak van een paar beelde, elk met 'n resolusie van 640 x 480.

Acknowledgements

My sincerest gratitude goes to the following people for helping to make this work possible:

- Dr. Kristiaan Schreve, my supervisor, for his constant encouragement and competent guidance.
- The members of the Machine Vision group, led by Prof. Ben Herbst and Dr. Karin Hunter, for freely sharing resources and ideas.

Above all, I thank God, His Son and the Holy Spirit, through whom all things are made possible.

Table of Contents

Declaration	i
Abstract	ii
Uittreksel	iii
Acknowledgements	iv
Table of Contents	v
List of Figures	ix
List of Tables	xi
Abbreviations	xii
Chapter 1 Introduction	1
1.1 Problem Statement.....	1
1.2 Project Context.....	1
1.3 Thesis Outline	2
Chapter 2 Literature Review	3
2.1 Optical Measurement Techniques	3
2.1.1 Passive Light Systems.....	3
2.1.2 Active Light Systems	4
2.2 Stereo Vision and Photogrammetry	5
2.3 Camera Calibration.....	6
2.3.1 Methods	6
2.3.2 Achieving Accuracy, Fast	9
2.3.3 Common Factors Influencing Successful Calibration	10
2.3.4 Calibration Object Design	12
Chapter 3 The Camera Model	16
3.1 Pinhole Model.....	16
3.1.1 Intrinsic Parameters.....	17

3.1.2 Extrinsic Parameters	20
3.1.3 The Camera Matrix	21
3.2 Additional Parameters	22
3.3 Lens Distortion Model	22
Chapter 4 The Measurement Process	25
4.1 Camera Calibration.....	25
4.1.1 The Method	25
4.1.2 Step 1: Initialisation of Camera Parameters.....	25
4.1.3 Step 2: Refinement of the Camera Parameters	26
4.2 Triangulation: Measuring Points in Three Dimensions	30
4.2.1 The DLT Method.....	30
4.2.2 Other Methods.....	30
4.3 Process Summary	31
Chapter 5 Image Processing	33
5.1 Software	33
5.1.1 The Python Scripting Language.....	33
5.1.2 The OpenCV Software Package.....	34
5.1.3 Data Visualisation.....	35
5.1.4 C++ Code in Python: the Wrapping Principle.....	36
5.1.5 Digital Video Camera Software.....	36
5.2 Automation	37
5.2.1 Pre-processing for Calibration	37
5.2.2 Point Correspondences in Two Images	38
5.3 Automated Detection of the Calibration Grid.....	38
5.3.1 Assumptions.....	38
5.3.2 Finding All Squares	39
5.3.3 Intermediate Steps	42
5.3.4 Deriving Sub-pixel Coordinates for Square Corners	43
5.4 Rapid Correspondence Matching	47

5.4.1 Tracking a Moving Laser Dot	47
5.4.2 Corner Detection Using Square Projections.....	49
5.4.3 Projected Line-Crossings.....	51
Chapter 6 Hardware	55
6.1 Digital Video Cameras	55
6.1.1 Camera Properties and Characteristics	55
6.1.2 Lenses.....	56
6.1.3 Communication with the Computer.....	57
6.1.4 Synchronisation	57
6.2 External Microcontroller	58
6.3 Laser Movement.....	58
6.4 Projector	59
6.5 The Calibration Object.....	60
6.5.1 Design	60
6.5.2 Manufacture	61
6.5.3 Measurement	61
Chapter 7 Experimental Setup and Planning.....	63
7.1 Positioning the Components.....	63
7.2 Illumination	64
7.3 Objects Used for Measurement	65
7.4 Definition of Error Measurements	65
7.4.1 Back-projection Error	66
7.4.2 Triangulation Error.....	66
7.4.3 Deviation from a Fitted Plane.....	66
7.5 Planning the Experiments.....	67
7.5.1 Variable Parameters and Variability.....	68
7.5.2 Correspondence Matching.....	68

Chapter 8 Experiments and Results	70
8.1 Variable Parameters and Variability.....	70
8.1.1 Base-to-depth Ratio.....	70
8.1.2 Camera Model Complexity	71
8.1.3 Variability.....	75
8.2 Correspondence Matching.....	75
8.3 A Practical 3D Measurement	78
Chapter 9 Conclusions and Recommendations	79
9.1 Conclusions.....	79
9.2 Shortcomings	79
9.3 Recommendations for Future Work	80
References	81
Appendix A Pseudo Code	86
A. 1. Sub-pixel Line Detection.....	86
A. 2. Correspondence Matching Using Corner Detection.....	88
A. 3. Correspondence Matching By Tracking a Moving Laser Dot.....	91
A. 4. Automatic Detection of Calibration Grid Corners.....	94
Appendix B Test Results	104
B. 1. Base-to-depth Ratio.....	104
B. 2. Camera Model Complexity.....	105
B. 3. Planar Deviation.....	107

List of Figures

Figure 2-1: Common shapes and patterns used for calibration objects.....	14
Figure 3-1: Pinhole camera model.....	16
Figure 3-2: Image plane with a principal point offset.....	18
Figure 3-3: Transformation from world to camera coordinate frame	20
Figure 3-4: Types of radial distortion	23
Figure 3-5: Radial distortion explained	24
Figure 4-1: Distribution of image and back-projected coordinates	27
Figure 4-2: Flow-diagram for optimisation function	28
Figure 4-3: Summary of Measurement Process	31
Figure 5-1: Top view of calibration grid and image plane.....	39
Figure 5-2: Camera views of calibration grid.....	40
Figure 5-3: Stepwise output of initial square-finding process	41
Figure 5-4: Simplified representation of calibration grid	42
Figure 5-5: Kernel with five elements used for 1D convolution	43
Figure 5-6: Stages in deriving accurate corner locations	44
Figure 5-7: 3D representation of intensity images	45
Figure 5-8: Illustration of edge extraction method.....	46
Figure 5-9: Laser dot on flat surface.....	48
Figure 5-10: Laser-tracking process	48
Figure 5-11: Projection of squares for automatic correspondence matching.....	50
Figure 5-12: Lines projected on flat surface.....	51
Figure 5-13: Derivative images of lines projected on flat surface.	52
Figure 5-14: 5x5 Sobel operator	52
Figure 5-15: Sum of derivative images	53
Figure 5-16: ROI around maximum intensity value when 5x5 kernel is used	54
Figure 6-1: Two-axis laser platform	59
Figure 7-1: Measurement system setup: schematic top-view	63
Figure 7-2: Actual measurement system setup.....	64
Figure 8-1: Back-projection errors for different camera models	73
Figure 8-2: Triangulation errors for different camera models	74

Figure 8-3: Error histograms and 3D visualisations for matching methods	77
Figure 8-4: 3D Visualisation of scanned bottle profile	78

List of Tables

Table 6-1: Certainty of measurement for calibration object corners	62
Table 8-1: Back-projection errors for varying base-to-depth ratios.....	71
Table 8-2: Triangulation errors for varying base-to-depth ratios	71
Table 8-3: Back-projection errors for different camera model complexities.....	72
Table 8-4: Triangulation errors for different camera model complexities.....	72
Table 8-5: Back-projection errors for variability study of calibrations	75
Table 8-6: Triangulation errors for variability study of calibrations	75
Table 8-7: Comparison of matching method accuracy.....	76

Abbreviations

2D	Two Dimensional
3D	Three Dimensional
A/D	Analogue to Digital
CMM	Computer Measurement Machine
CMOS	Complimentary Metal-Oxide Semi-conductor
CCD	Charge Coupled Device
DLP	Digital Light Processing
DLT	Direct Linear Transformation
DMD	Digital Micromirror Device
fps	frames per second
GUI	Graphical User Interface
ROI	Region Of Interest
RMS	Root Mean Square
SVD	Singular Value Decomposition

Chapter 1 Introduction

1.1 Problem Statement

This project's overall goal is the development and implementation of a rapid optical measurement system using digital video cameras. It is to be a first step in developing a complete measurement system capable of quality control for relatively small, mass-produced (and possibly deformable) objects such as plastic bottles.

As a first step for a more advanced system, there are a few requirements that must be met. Firstly, a basic working measurement system must be established consisting of relatively inexpensive hardware components. These components must be fully reusable and reconfigurable in future developments. The system must secondly be free from software licence constraints, but not only to keep the development cost down. The software used must also be of such a nature that it allows opportunity for commercialisation of any software developed for the system. Thirdly, the system must be as accurate as possible without interfering with the fourth requirement. This fourth requirement is that an understanding must be established of the underlying principles governing an accurate and rapid measurement system. The use of more accurate methods that are freely available as software packages might have to be sacrificed in order to achieve this by implementing certain processes from basic theory. The fifth and final requirement is that the whole measurement processes must be automated as far as possible to achieve rapid measurement while maintaining flexibility.

1.2 Project Context

Optical measurement techniques have traditionally been bound to specific applications requiring expensive and specialised equipment. With the rapidly developing digital technologies in the market, computers and off-the-shelf digital cameras are continually improving in both speed and capability while also becoming less expensive. This in turn has made optical measurement techniques not only more accessible in terms of cost, but has also enabled new or alternate solutions to common problems.

The inherent characteristics of an optical measurement system allow it to make non-intrusive measurements. This includes measuring surfaces with smooth curvatures that cannot be measured using devices such as micrometers. While touch-probe devices provide very good accuracy, they are usually slow, large and

very expensive. Using an intrusive technique, they also cannot be used for deformable objects, such as foam prints.

This project is an extension of a final year project (Van der Merwe, 2005) that used a high resolution digital camera for a simple stereo-vision measurement. Here the work is taken further, but for a stereo pair of digital video cameras. It will present an inexpensive alternative to the touch-probe technique for applications that do not require such excessive accuracy, but rather rapid measurement and assessment.

1.3 Thesis Outline

The following chapter will cover the literature applicable to this project to establish what techniques are available and how certain factors will influence the requirements of the project.

The basic theory and mathematical models used for the project are then covered, followed by an explanation of how the theory is implemented specifically for this project. This is followed by a detailed chapter on the image processing used to automate the whole process and achieve accuracy. The hardware components and their applicable characteristics are then discussed. The third and second to last chapters present the experimental setup and subsequent results. These chapters show that measurement accuracies below 0.4 mm (for a 235 x 190 x 95 mm volume) can be reached using the simple techniques presented. It also shows that data-sets of thousands of measurements can be made within minutes using the automated and semi-automated processes of calibration, coordinate extraction and stereo-matching developed for the system.

The final chapter gives conclusions and recommendations, also evaluating the outcomes and shortcomings of the project.

Chapter 2 Literature Review

With the wide range of literature available on the subject of non-intrusive measurement, the literature review will focus on measurement techniques that use digital cameras as their main data receiver component.

As far as applications are concerned, the focus of this chapter will be on techniques lending themselves to accurate metrology (Fraser *et al.* 1995; Muller *et al.* 2007; Valkenburg & McIvor, 1998; Pappa *et al.* 2000). Other applications, ranging from real-time facial measurement (Zhang & Huang, 2006) to time-consuming modelling of full-scale statues (Guidi & Atzeni, 2004), can also be found. These are, however, either focussed on visual quality rather than accuracy or too time consuming.

Following a review of the available techniques, the two main directions, or rather, approaches driven by different focus areas in vision metrology will be addressed. The bulk of the literature review then covers the topic of camera calibration, because it plays a definitive role in the methods that can be used for a measurement system as well as the achievable accuracy.

2.1 Optical Measurement Techniques

There are many ways in which optical measurement techniques could be classified: the specific application, speed, accuracy or assortment and type of components used. In this case the latter criterion will be used to differentiate between methods using either active or passive light sources.

2.1.1 Passive Light Systems

For these techniques, the light-source plays no active role in the calibration, measurement or feature detection process, except for providing general illumination on the object. Such systems usually consist of a single camera capturing multiple images or multiple cameras rigidly mounted with respect to one another, each capturing a single image.

In the single camera case, the movement of the camera or object is usually constrained in some way, such as an object undergoing pure rotation on a turn-table (Jiang *et al.* 2004; Fitzgibbon, 1998). More general camera or object motions are also allowed (Hao & Meyer, 2003; Luong & Faugeras 1997), but care has to be taken to avoid certain critical or fatal motion sequences (Hartley & Zisserman, 2003: 497; Ma *et al.* 2004: 293). One advantage, however, is that some of these methods allow the textured colour reconstruction of objects (Elter *et al.* 2007). The methods

also allow the reconstruction of a large number of coordinates. In all these cases, however, easily identifiable features are needed for points to be matched in multiple images. This is the greatest disadvantage of these methods: the reconstruction is at the mercy of optically cooperative surfaces, with easily identifiable features, such as textures. To overcome this problem, some passive light systems use object silhouettes under rotation (Esteban & Schmitt, 2003) or just silhouettes at different angles under more general movement (Boyer, 2005). Again, the accuracy (or lack thereof) prohibits the use of such methods in the context of this project. To achieve accuracy, easily identifiable and well contrasted markers can be introduced (Fraser *et al.* 1995; Pappa *et al.* 2000). These techniques do in reality use more than one camera, but the methods allow the use of only one camera capturing images at different angles. The markers allow very accurate location extraction of coordinates, but the number of measurements is then limited to the number of markers. They are also bound to time-consuming post-processing for the final measurement.

In the multiple camera applications, either markers (Muller *et al.* 2007; Pedersini *et al.* 1999) or motion detection (Schraml *et al.* 2007) can be used to identify point correspondences. In these cases, if the object is moving the cameras need to be synchronised in order to capture the same feature at exactly the same time. For common off-the-shelf cameras, such synchronisation is usually not possible and components that are more application specific would have to be acquired. The advantage of these techniques is that for every pair or set of points that are matched, the 3D coordinates can immediately be determined via triangulation. An initial camera calibration is usually needed and cameras have to be re-calibrated frequently to maintain accuracy.

2.1.2 Active Light Systems

Active light systems will be classified into two categories: those playing a part in the calibration procedure and those who do not. The greatest advantages of these techniques are that they enable image coordinates to be extracted accurately and in large numbers. The active light projections also enable correspondences in multiple images to be easily identified and matched.

Calibrated Light Sources

For these techniques, the position or geometry of the light source itself or the light-source pattern needs to be included in the calibration process.

In a number of methods the active light source (usually a DLP projector) is treated just like a camera that needs to be calibrated (Valkenburg & McIvor, 1998; Guisser *et al.* 2000; Zhang & Huang, 2006). This is possible because the projector

has many of the same physical properties as a camera, only the light rays are projected from it and not into it. See section 2.3 for more on calibration.

Using a completely different approach, some techniques make use of a phase-shifting principle (Chi-Fang & Chih-Yang, 1999; Quan *et al.* 2001; Zhang *et al.*, 2002; Zhang & Huang, 2006). In these methods, a light source (laser, DLP projector or DMD device) is used to project sinusoidal intensity patterns onto an object. Each pattern is out of phase by a known number of degrees. Certain unknowns have to be calibrated for the system by typically moving a reference surface through a known distance and projecting the phase patterns on the surface after each movement. The main advantages for these techniques are that they can acquire large numbers of 3D measurements (complete depth maps for every pixel coordinate in an image) and at high speeds.

Uncalibrated Light Sources

For these methods the active light source is simply used as means to solve the correspondence matching problem for images from different angles. An exception is found in the case of Scharstein & Szeliski (2003), who only use one camera and an active light-source.

Gühring (2000) combines multiple grey-code patterns with a line shifting sequence to detect correspondences in a stereo camera setup. The cameras are calibrated beforehand using multiple images of a planar pattern.

It is of course also possible to use laser dots or lines to solve the correspondence problem by scanning them across any arbitrary surface, but this limits the speed with which coordinates can be acquired. They do however have the advantage of being depth-invariant in contrast with the methods using DLP projectors that are only in focus for a specific depth.

2.2 Stereo Vision and Photogrammetry

Even though based on the same working principles and even the same mathematical models, there is a notable difference between stereo vision and photogrammetry. The latter finds its origins in the measurement of landmasses for cartography. Very expensive cameras with specialised lenses and equipment is mounted on an aeroplane for measuring landmass regions, hence the name aerial photogrammetry. This field of metrology established the basic camera models and mathematics used for calculating object depth from two images (also known as a stereo pair). In aerial photogrammetry, consecutive overlapping images are used as stereo pairs for calculating depth information.

With inexpensive digital cameras flooding the market, the same principles used in photogrammetry found its way into the field of computer vision, or more specifically, stereo vision. Where expensive photogrammetric measurement systems must adhere to certain standards of excellence concerning accuracy and methodology, many (but not all) computer vision applications tend to forego these standards. This is because many of the machine vision applications do not require nearly the same level of accuracy and are sometimes more concerned with the visual quality of a 3D reconstruction than its quantitative accuracy.

With so many applications now being made possible in optical measurement, the challenge remains to somehow achieve levels of accuracy comparable to classical photogrammetric techniques. For this to be done while still maintaining the advantages provided by off-the-shelf components not dedicated to photogrammetric application is not a simple task.

2.3 Camera Calibration

Camera calibration is the determination of the unknown camera parameters that describe the mathematical camera model. These parameters are needed in order to measure depth using only 2D image information.

Camera calibration is one of the most important steps in the measurement process, because it directly influences the achievable accuracy of the measurements. Even though it is not the only influence on accuracy, it acts as a potential bottleneck for the final accuracy of the measurement system.

This section is dedicated to differentiate between the myriad of available techniques and focuses on those with greatest relevance to this project.

2.3.1 Methods

Some important calibration methods will now be discussed with the focus on their accuracy and also their practical application with respect to the type of control points needed in multiple images. Control points are any features, such as reflective markers, used to extract image coordinates for calibration. These control points can have known or unknown world coordinates depending on the calibration method. This discussion is used to aid in the final design and implementation of the measurement system of this project.

Many techniques are available that will not be discussed because they are not accurate or consistent enough, making them impractical for use in this project. These methods include calibration from object shadows (Cao & Shah, 2005), using object silhouettes (Boyer, 2005), objects under circular motion (Zhang, 2006) or image

sequences using a single moving camera (Hao & Mayer, 2003; Luong & Faugeras, 1997).

Worthy of mention before the methods are discussed is the topic of bundle-adjustment. Mikhail *et al.* (2001:123) claims bundle-adjustment to be the most accurate method of triangulation in use, but involves more unknowns than other triangulation methods. It is consequently computationally intensive, not lending itself to rapid measurement. Having focussed mainly on the faster and simpler, yet accurate calibration methods from the machine vision side, the implementation or use of bundle-adjustment falls outside the scope of this project. It will be clearly mentioned if bundle-adjustment is used in any of the case studies discussed from the referenced literature in order to separate these cases from other calibration methods using a machine vision approach.

Self-calibration

Self-calibration does not require that control points in images have known coordinates, eliminating the need for an accurate calibration field or object. As stated by Brown (1972), a “satisfactory” calibration is possible without the use of any control points, referring to points with known world coordinates.

Thus far the author has only found one case of self-calibration for digital cameras in the literature (Fraser *et al.* 1995) that achieves accuracies that are comparable with classic film-based photogrammetry. From the machine vision arena, the final measurement accuracy of self-calibration methods found (Luong & Faugeras, 1997; Foroosh *et al.* 2005) is considerably less than achieved by Fraser. It must be noted that there are many factors influencing the final accuracy of each method and that there is no official measurement standard by which these methods can be compared. The focus of these studies is also not the same, but most importantly, Fraser uses a bundle-adjustment technique where the others do not.

To achieve the type of accuracies reported by Fraser *et al.* (1995), the bundle-adjustment method requires a large number of control points that are well distributed throughout the measurement volume (Fraser used 120 markers). The location of each point in an image must also be extracted with very high accuracy (~0.03 pixels) while multiple images (30 – 100) from a range of angles must be acquired. Theoretically, this method only needs three different views of the control points if the internal parameters of the camera stay constant (no zooming or change in focus).

Calibration Using 2D Calibration Objects

The main advantages of methods using flat calibration objects are that they are relatively easy to manufacture and to use in mobile applications. Using multiple images of a printed pattern on a flat surface at varying distances from the camera (section 2.3.3) aids in accurate calibration for many methods. It also increases the

effective volume that can be used for accurate measurement. If the factors influencing calibration are not properly understood, however, calibrations can be made to yield much greater errors than expected.

The ease of manufacturing (where a pattern is usually just printed on a piece of paper) can also be a disadvantage. It is difficult to verify the pattern's accuracy, because it usually requires some other visual measurement technique.

There are a number of calibration methods that make use of a planar calibration object with some easily identifiable patterns or geometries (Tsai, 1987; Pedersini *et al.* 1999; Fremont & Chellali, 2002; Cao & Foroosh, 2004; Zhang, 2000; Triggs, 1998; Xue *et al.* 2007; Batista *et al.* 1998).

Some of these are novel in their use of a specific pattern geometry, such as large circles (Fremont & Chellali, 2002) or an isocetes trapezoid (Cao & Foroosh, 2004). Most methods, including those most cited and studied in the machine vision community, use either circular or square features (Tsai, 1987; Triggs, 1998; Zhang, 2000). Some of these methods have been compared and results given on the final accuracy in different formats (Armangué *et al.* 2002; González *et al.* 2005). From these studies it can be seen that one of the oldest methods, that of Tsai (1987), achieves the best overall triangulation accuracy. With so many variables in the calibration setup, it cannot be said for certain if Tsai's method will perform the best with regards to triangulation under all circumstances.

Worthy of note is the simplicity of Tsai's camera model. It only contains one radial distortion coefficient, ignores decentring (tangential) lens distortion and pixel skew and assumes that the optical centre lies exactly in the middle of the image centre.

Calibration Using 3D Calibration Objects

3D calibration objects have a few practical disadvantages over the 2D objects: they are more difficult and expensive to manufacture and they are not as easy to transport for use in mobile applications. Another disadvantage is that the size of the calibration object limits the volume in which accurate measurements can be made. The 2D patterns are more flexible in this regard. As apposed to 2D objects, the more advanced manufacturing methods needed for 3D objects also present a certain advantage. For instance, if blocks or spheres are used it can aid in very accurate measurement of the object features using touch-probe measurement techniques. More is said about this and the design of such objects in section 2.3.4.

Because the computer vision community tries to move towards less expensive solutions, it is not surprising that there are not many methods using 3D calibration objects. Two methods have been found that are worthy of mention.

The first is that of Tsai (1987), which is the same one discussed in the 2D calibration object section. It is not only an accurate calibration method in the 2D case, but also versatile in its ability to use 3D objects as well. Only one practical

application was found in the literature so far using this method (Muller *et al.* 2007). Muller uses an added step for estimating the lens distortion which includes an extra distortion coefficient and a drifting radial centre. This makes it difficult to evaluate the accuracy of Tsai's method separately. Even though the final triangulation results are good, they are not given in a format directly comparable with other studies.

The second method is that of Heikkilä (2000) that uses a 3D object with circular markers. Heikkilä & Silvén (1997) have added implicit image correction. In both cases, the bias produced by circular feature location has been compensated for. When compared to Tsai, Heikkilä's camera model is more complex: it includes a second radial distortion parameter as well as two more parameters for tangential distortion.

Only one comparative study was found for these two methods (Remondino & Fraser, 2006), in which Tsai and Heikkilä's methods were both implemented using the same images of a 3D grid. Curiously enough, Tsai's method with the simpler distortion model still performs better than Heikkilä's method. The study also compared these two methods with three other visual metrology packages (PhotoModeler, Australis and SGAP) that use bundle-adjustment. Even though the image errors were of the same magnitude for all the techniques, the bundle-adjustment packages clearly yielded the most accurate triangulation results. Important to note is that the method and the accuracy with which the calibration object was measured is not given in the study.

2.3.2 Achieving Accuracy, Fast

Discussed here are the principles that make it possible for the previously mentioned techniques to achieve accuracy without the computational intensity needed for bundle-adjustment methods.

The main difference when compare to bundle adjustments is not only that there are usually less parameters to be estimated, but also that approximate solutions for linear parameters can be determined with great speed. This is done using linear or closed-form solutions such as the DLT algorithm (Hartley & Zisserman, 2005: 88) or such as those proposed by Csurka *et al.* (1998). These methods ignore the non-linear effects such as lens-distortion, making use of linear algebra techniques such as SVD to solve sets of equations. The equations are based on the relatively simple relation between a set of known world coordinates and their corresponding image coordinates. This is another reason why calibration objects are needed.

Even though these linear methods are not sufficient on their own to achieve the necessary accuracy needed for metrology applications, they can usually approximate good initial values. These values can be passed on to the next step in the process: optimisation.

All of the techniques that can potentially be used for metrology applications (Zhang, 2000; Triggs, 1998; Heikkilä & Silven, 1997) make use of an optimisation routine (or multiple routines) after the calculation of an initial guess. Because of the good initial values, these routines usually converge quite fast. They can differ in their order and mathematical application, but somewhere along the line they make use of a standard optimisation algorithm such as Levenberg-Marquardt.

A common variable to minimise in these routines is the back-projection error, which is discussed in section 7.4.1.

2.3.3 Common Factors Influencing Successful Calibration

Many of the methods discussed in the previous section have some common factors that influence the accuracy and success of the calibration. In certain cases the factors are essential, while in others it is simply advantageous. A set of criteria was already formulated almost two decades ago for the successful self-calibration case (Brown, 1972), but is still applicable to most applications of the methods using 2D or 3D calibration objects. These criteria are summarised by Clarke & Fryer (1998), with a more comprehensive summary based on a number of studies given by Remondino & Fraser (2006). Remondino & Fraser and Brown's criteria will be combined and discussed, acting as a guide for the design of the project's measurement system.

Number of Rays

This criterion refers to the number of times the same control point is in different images, each captured from a different angle. The point projected through the camera centre onto the image plane (see section 3.1) forms the ray. For the self-calibration case, at least three views of the same point is necessary. For the other two cases, an increased number of views will usually cause greater accuracy, up to about eight rays (or views) per point.

Angles of Convergence

With an increase in angles between rays formed by the same point, the accuracy of the calibration network will also increase. The practical implication is that the "base-to-depth" ratio should be as large as possible. The base refers to the distance between camera centres and the depth refers to the perpendicular distance from the base-line to the point being measured. This is applicable to all the calibration methods.

No studies were found so far to give an idea of what the accuracy increase would be as the ratio increases.

Amount and Distribution of Points

The calibration accuracy increases as more points are measured per image. Tsai (1987) developed a method for determining the number of points needed. As a rule of thumb, anything “more than a few tens of points” should suffice (Remondino & Fraser, 2006). In Tsai’s simple camera setup with only two cameras, 60 points produced good results.

Apart from having a sufficient number of points, they should also be well distributed throughout the 3D volume that is finally used for measurement. The parameters estimated by the system can be expected to achieve accurate measurement only for coordinates within the same volume in which the calibration points were distributed (Pedersini, 1999). This applies not only to the self-calibration case, but has practical implications for the other methods as well.

In the case of 2D patterns where multiple images are captured for calibration, the pattern should be moved to different object distances. When using a rigid 3D grid, it should be designed large enough to fill the volume in which objects are to be measured.

Orthogonal Roll Angles and Projective Coupling

Projective coupling refers to the correlation between the internal and external camera parameters. An example given by Shortis *et al.* (1995) is the typical coupling between the principal point location, decentring distortion and the tip or tilt of the camera. Small changes in any of these parameters will still yield the same overall calibration result.

This coupling can have both advantages and disadvantages for calibration. For successful self-calibration, the criterion stipulates that this coupling effect must be “broken”. This can be done by capturing images after rolling the camera orthogonally with respect to previous image acquisitions. A minimal requirement in self-calibration is that at least one image must be “rolled” by 90 degrees with respect to the others if only three images are captured. It is not clear whether this breaking of the projective coupling aids in the convergence of the optimisation problem for self-calibration. It does, however, effect the choice of method used for calculating 3D structure.

In robotic applications, where the camera is mobile with respect to the world coordinate system, the 3D structure calculation uses the constant internal parameters acquired via calibration along with point correspondences in multiple images. With a strong projective coupling during calibration, the internal parameters cannot be accurately separated from the external parameters. This can cause subsequent errors in 3D calculations to be much greater than anticipated by the initial calibration. Without actually addressing projective coupling, Boufama & Habed (2004) illustrates how “noisy” internal parameters can still yield relatively good 3D

structure results. It is noticed in their study that this is achieved by using proper numerical conditioning and, for their best results, enough point correspondences.

As an advantage, the projective coupling effect can compensate for variations within the linear section of the distortion curve if only a partial field of view is used in the camera lens (Fraser *et al.* 1995). For the case where there is a strong projective coupling, Remondino & Fraser (2006) as well as Tsai (1987) makes a similar observation: there is a negligible difference in the final 3D accuracy if the principal point offset parameters are given different values (within a reasonable range). Remondino & Fraser notes this is also true for the decentring distortion terms. The stability of external parameters for varying internal parameters has also been reported by González *et al.* (2005) in a stability study of a number of calibration methods.

In general, projective coupling is advantageous if the cameras are rigid and all final calibration parameters are used in combination to calculate 3D structure for the specific volume spanned by the calibration field. As mentioned, strong coupling can also be disastrous, rendering the calibration almost useless if the internal parameters are to be used independently of the scene geometry and camera orientation.

2.3.4 Calibration Object Design

Based on the previous discussions on calibration, it is assumed that some kind of calibration object will be used in the calibration process. The advantage of using such an object is twofold: barring extensive non-linear effects, it allows for a good initial guess of the camera parameters using simple linear calibration techniques. These parameters can then be passed on to an optimisation routine to calculate additional parameters for a more accurate camera model. Secondly, if a calibration object can be accurately measured, the known coordinates of its features can be compared to the triangulated coordinates of the same features after calibration. This can then give an immediate statistical measurement of the system's achievable accuracy, which is important in the scope of this project. It can also aid in future development of more accurate calibration techniques.

The initial measurement of the calibration object, however, can in itself be a disadvantage. Depending on the type of optimisation used, the accuracy with which the calibration object is measured can limit the achievable accuracy with which the camera parameters are determined. This aside, the aspects influencing both practical implementation and final accuracy will now be discussed.

Feature Detection and Location

One of the most important things to consider when designing a calibration object is the accuracy with which known feature coordinates can be extracted. In general, the

greater the accuracy with which a feature is extracted, the greater the accuracy of the calibration. According to Mallon & Whelan (2006), some calibration methods (Strum & Maybank, 1999; Zhang, 2000) assume that feature coordinates are extracted with zero mean Gaussian distributions for the optimisation procedure to converge to an optimum solution. Even if such high image coordinate accuracy is not needed for accurate calibration, the triangulation accuracy of a coordinate will be directly influenced by the accuracy with which the corresponding point in a stereo image pair is extracted.

Before the location of a feature can be determined, the other important consideration is the initial recognition of the features in an image. From an image processing point of view, the simplest way in which to aid automatic detection is by using high contrast features (Shortis *et al.* 1994). Examples of this would be markers made of reflective material that can be used for either the calibration object as implemented by Muller *et al.* (2007) or simply for matching corresponding coordinates in multiple images as implemented by Pappa *et al.* (2001). High contrasted black and white patterns can also be used, in some cases being a simple pattern printed on paper. Using simple geometric shapes for the features, such as circles, squares or corners, can then further aid in the recognition phase by removing objects that may be well contrasted, but do not fit the geometric criteria.

Choosing the Pattern Geometry

To add to the previous section on feature location, it is necessary to also discuss the type of shapes that can be used in the calibration object design. In image processing, a number of commonly implemented methods are used for the accurate sub-pixel extraction of target locations. This should be kept in mind when designing the calibration object, because the methods are dependent on specifically shaped features. In the case of a 3D calibration object, this could (along with the contrast requirement) even dictate the manufacturing processes that would be used.

A few of the commonly used shapes and patterns that enable accurate target location include circles or spheres, rectangles and checkerboard patterns. Figure 2-1 shows the basic shapes and the possible patterns, keeping in mind that they are not restricted to two dimensions, as in the case of the circles that can also be spheres. For each of these shapes a different image processing method is used to extract accurate target locations. For the rectangles or checkerboard patterns in (a) and (b), corners can be initially detected using, for instance, Harris corner-detection. At the cost of extra computation, sub-pixel refinement of the corner locations can then be made using interpolation between pixels (Ma *et al.* 2004: 379).

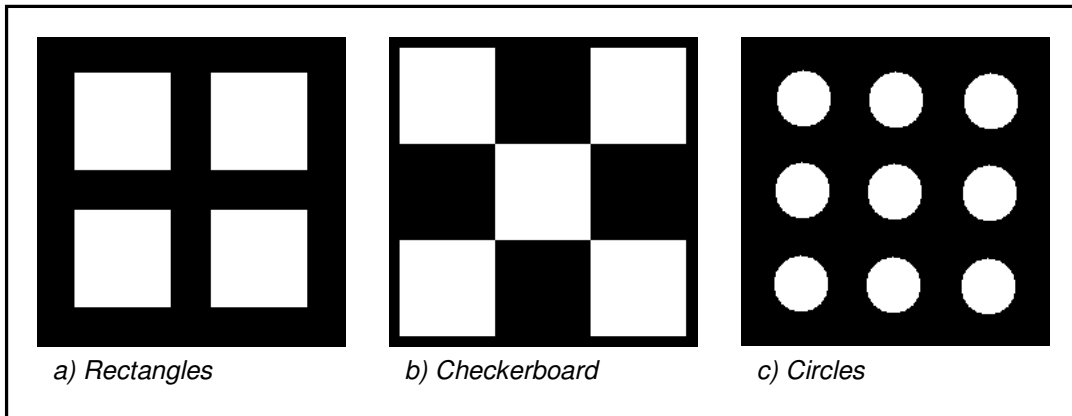


Figure 2-1: Common shapes and patterns used for calibration objects

Another method of refining the corner coordinates in these two cases is by using edge information to calculate line intersections, as demonstrated by Tsai (1987). Mallon & Whelan (2007) briefly discusses this method, as well as corner refinement using surface fitting to the corner's intensity profile. For circular features a number of locating methods are discussed and evaluated by Shortis *et al.* (1994).

The accuracy with which the coordinates of each of these shapes can be extracted using their corresponding methods is influenced differently by lens distortion and perspective effects of an optical system. Mallon & Whelan (2007) found that circular patterns yield the least accurate target location, being influenced by the lens distortion as well as the perspective effects. The best results were found for the line-intersection method which is invariant under perspective transformation, but is still influenced by lens distortion. Even so, this method can be more accurate than the corner refinement method if lens distortion is moderate.

Verifying the Accuracy of the Calibration Object

To reiterate, the error analysis of the calibration grid's triangulation results would be a useful first indication of the system's achievable measurement accuracy for that specific calibration. In order to gain this analytical advantage, it must be made possible for the calibration object to be measured with high accuracy. The practical implication of this is that planar patterns (such as those printed on a piece of paper) cannot be used easily. Only one article was found in the literature that verifies the accuracy of the planar pattern (Pedersini *et al.* 1999) and this was by means of a classic photogrammetric procedure claiming an accuracy of "better than 0.1 mm". The problem with this is that the achievable measurement accuracy of the system itself is claimed to be "better than 0.2 mm", which leaves a 0.1 mm uncertainty based on the photogrammetric measurement. These results do, however, indicate the measurement accuracy that can be expected of such a system. The accuracy with

which objects are to be measured in the scope of this thesis is therefore expected to be well below 1 mm.

It is deemed important in the scope of this project to verify the certainty with which the calibration object is measured in order to effectively evaluate the optical system's measurement results. Section 6.5.3 deals with the measurement of the calibration object.

Chapter 3 The Camera Model

3.1 Pinhole Model

The simplest mathematical description for a camera is the pinhole model, also known as the perspective camera model. Most camera calibration methods found in the literature use the pinhole model as one of their first and most basic assumptions. The pinhole model is in turn derived from the idealised optical properties of a thin lens. The thin lens model neglects physical thickness and is only concerned with the radii of its surfaces (Mikhail *et al.* 2001). The basic properties of the thin lens model are used in the field of photogrammetry to derive the collinearity equations, which is equivalent to the equations used in machine vision for stereo measurement. Thick lenses such as those found in real cameras can be modelled by calculating a mathematically equivalent thin lens (Mikhail *et al.* 2001) and will be a good approximation of a well-focused imaging system (Ma *et al.* 2004). Using the equations based on the pinhole model, the calculation of a thin lens equivalent is achieved automatically as part of the calibration process.

Figure 3-1 illustrates the projection of a world coordinate \mathbf{P} onto the image plane for the pinhole model. According to the thin lens properties, the image point, \mathbf{p} , must lie on the intersection of the straight line (formed by \mathbf{P} and \mathbf{C}) and the image plane, \mathbf{L} .

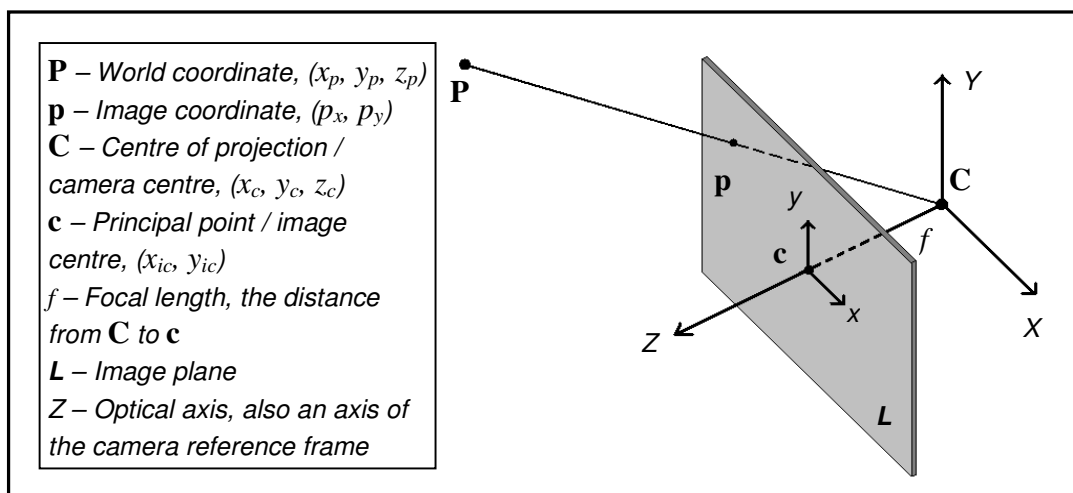


Figure 3-1: Pinhole camera model

The camera coordinate frame is orientated with the centre of projection (or camera centre), \mathbf{C} , as origin of the X -, Y - and Z -axes as shown. The Z -axis is perpendicular

to the image plane, intersecting it at the principal point, \mathbf{c} . The principal point is also known as the optical centre or image centre and forms the current origin for the image reference plane with the x - and y -axis as shown.

Note that this illustration might be confusing at first, because the image plane in a real camera is behind the centre of projection, \mathbf{C} , causing the image to be projected upside down. In Figure 3-1, the image plane is shifted in front of \mathbf{C} by the focal-length distance f instead of behind \mathbf{C} . The image is still geometrically the same, but will now be displayed the right way up, which is more convenient.

Using the pinhole model as the first building block, other physical effects such as lens distortion or skew pixels can then be added to get a more accurate approximation of a real camera.

The next three sections, however, will first show how the mathematical model enables the projection (or mapping) of a world coordinate point in an arbitrary coordinate frame to the image plane of a digital camera. This will eventually enable the triangulation and measurement of world coordinates from a pair of images.

3.1.1 Intrinsic Parameters

There are two sets of parameters needed to achieve the projection of an arbitrary world point onto the image plane of a digital camera device. The first set of parameters describes the internal geometry of the camera. These are called the intrinsic parameters and they stay constant if the camera goes through an arbitrary translation and rotation. The second set is the external or extrinsic parameters, defining the rotation and translation transformation of the camera from the world-coordinate frame to the camera-coordinate frame.

Note that the calibration matrix described in the next section is in terms of a retinal-plane coordinate frame measured in metric units. This would be equivalent to a film camera that is measured in (for instance) millimetres. The calibration matrix used for a digital camera (discussed after the one for a film camera) includes information about the pixel elements in the sensor-array.

Camera Calibration Matrix

The intrinsic camera parameters can be written in the form of the camera calibration matrix given in Equation 3-1.

$$\mathbf{K}' = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Equation 3-1}$$

Referring to Figure 3-2, the origin of the image plane does not have to coincide with the principal point, \mathbf{c} . The digital images used in this project, for instance, all have their origin in the top left corner, with the axis in the directions as shown. In order to take into account this offset of the principal point, the calibration matrix contains the p_x and p_y terms. These values are the positive distances from the new origin to the principal point, \mathbf{c} .

Now, if the world-coordinate frame is set with its origin at the camera centre, \mathbf{C} , and its axis as shown in Figure 3-1, but with the Y-axis in the opposite direction, then a mapping of the world point, \mathbf{P} , to the image plane is possible.

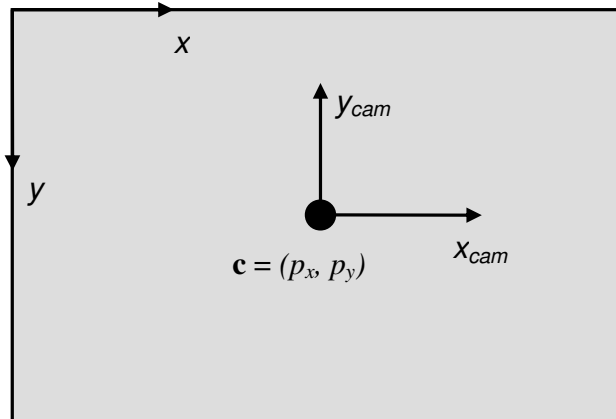


Figure 3-2: Image plane with a principal point offset

This is illustrated by Equation 3-2, where the last column vector contains the world coordinates that are to be projected onto the image plane, written in homogeneous form.

For homogeneous coordinates an extra value (1 in the case of finite points and lines) is added to the end of the coordinate vector. This notation allows for points and lines at infinity to be represented. The first column vector in Equation 3-2 contains the projected image coordinates, also in homogeneous form. Equation 3-3 shows the compact matrix notation of Equation 3-2.

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \quad \text{Equation 3-2}$$

$$\mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{I} & \mathbf{0} \end{bmatrix} \mathbf{X} \quad \text{Equation 3-3}$$

The question might now arise: if the projection was accomplished with only the intrinsic parameters, why are the extrinsic parameters still needed?

The projection was only possible in this case because the world coordinate frame was set to the camera centre. This in turn is only possible if the position of the camera centre is known relative to the world coordinates being projected. If some arbitrary coordinate frame is used, a rotation and translation will have to be added as defined in section 3.1.2.

The Calibration Matrix for a Digital Camera

In a digital camera, the physical equivalent of the image plane consists of an array of pixel elements. The previous section only described a retinal plane coordinate frame such as for a film camera. A mathematical relationship between the pixel array and retinal plane coordinate frame must now be established.

Equation 3-4 shows the camera calibration matrix for which the pixel elements have been taken into account.

$$\mathbf{K} = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{Equation 3-4}$$

The first difference in this version of the calibration matrix is the parameter s which is also called the skew factor (Ma *et al.* 2004). This parameter allows for pixels that do not form square angles, but is set to zero for all but a few unusual cases (Hartley & Zisserman 2003).

Besides the skew factor, the more important difference is that each of the matrix entries incorporates the width and height of the pixels. Looking again at the entries in Equation 3-1, the focal length terms in Equation 3-4 become $\alpha_x = fm_x$ and $\alpha_y = fm_y$, while the principal offset values become $x_0 = m_x p_x$ and $y_0 = m_y p_y$. In each of these conversions, the m_x and m_y values are the pixel width and height respectively given in the number of pixels per metric unit. Multiplying them with the entries in Equation 3-1 that are in metric units, the entries of the new calibration matrix are expressed in terms of pixels. If the pixels are square then m_x and m_y will be equal and the new focal length terms should have the same value. For most cameras the pixels are very nearly square.

A good way to test whether a calibration matrix is valid is by seeing if the two focal length terms on the diagonal are more or less the same and whether the principal point values are more or less in the middle of the image. The calibration matrix as used in the rest of the project is the same as the one in Equation 3-4, except for the skew parameter which will be assumed to be zero.

3.1.2 Extrinsic Parameters

A world coordinate frame can be established by using, for instance, some known geometry in a scene. In order to project a point to the image plane, the camera centre's position and orientation as well as the coordinates of the point must be known in the established coordinate frame. A translation and rotation is needed to transform the world coordinate frame to the camera coordinate frame as shown in Figure 3-3.

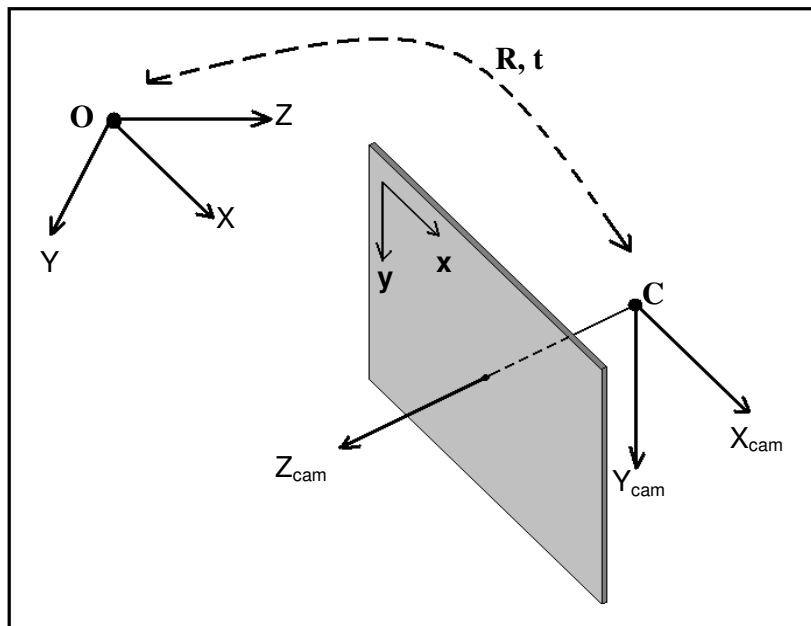


Figure 3-3: Transformation from world to camera coordinate frame

The rotation and translation matrices, \mathbf{R} and \mathbf{t} , and the camera centre, \mathbf{C} , relate the camera position and orientation to the world coordinate frame. Equation 3-5 shows the rotation matrix used for the orientation transformation. Mikhail *et al.* (2001) shows how this rotation matrix is constructed from three single rotation angles around each axis of the coordinate frame. The rotation matrix could also be expressed as a more compact vector form containing only three entries (Ma *et al.* 2004). Equation 3-6 is simply the Euclidean world coordinates of the camera centre.

$$\mathbf{R} = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \quad \text{Equation 3-5}$$

$$\mathbf{C} = \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \quad \text{Equation 3-6}$$

The next section shows how these extrinsic parameters are used in combination with the intrinsic parameters to achieve the projection to the image plane.

3.1.3 The Camera Matrix

Equation 3-2 shows how a world coordinate is projected onto the image plane if the world coordinate frame is already set to the position of the camera centre. For an arbitrary world coordinate frame the knowledge of the extrinsic parameters have to be added to make the necessary transformation. Equation 3-7 shows how the calibration matrix for a digital camera, \mathbf{K} , as well as the rotation matrix, \mathbf{R} , and the camera centre, \mathbf{C} , are used to project a world coordinate onto the image plane. The compact matrix notation is shown in Equation 3-8.

$$\begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha_x & 0 & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -x_c \\ 0 & 1 & 0 & -y_c \\ 0 & 0 & 1 & -z_c \end{bmatrix} \begin{bmatrix} x_p \\ y_p \\ z_p \\ 1 \end{bmatrix} \quad \text{Equation 3-7}$$

$$\mathbf{x} = \mathbf{KR}[\mathbf{I}|\mathbf{-C}]\mathbf{X} \quad \text{Equation 3-8}$$

The camera matrix is therefore expressed by Equation 3-9. To eliminate the identity matrix, the rotation matrix and camera centre can be combined as in Equation 3-10 to give Equation 3-11, which is another representation of the camera matrix. Multiplying these matrices gives the final 3x4 camera matrix.

$$\mathbf{P} = \mathbf{KR}[\mathbf{I}|\mathbf{-C}] \quad \text{Equation 3-9}$$

$$\mathbf{t} = \mathbf{-RC} \quad \text{Equation 3-10}$$

$$\mathbf{P} = \mathbf{K}[\mathbf{R}|\mathbf{t}] \quad \text{Equation 3-11}$$

Now that the camera matrix has been established, it can be used to map a world coordinate to the image plane as shown in Equation 3-12.

$$\mathbf{x} = \mathbf{PX} \quad \text{Equation 3-12}$$

3.2 Additional Parameters

With the camera matrix defined, additional parameters can now be added to increase the accuracy of the camera model. Clarke & Fryer (1998) define additional parameters as those besides the radial and tangential lens distortion that are simply added because it increased the accuracy of calibration. They also report that these additional parameters many times have “no foundation based on observable physical phenomenon” and that too many of these parameters could “weaken the solution for the coordinates of target points”. The skew factor defined in the calibration matrix (Equation 3-4) is a good example of an additional parameter based on a very clearly defined physical observation in a digital camera’s image array.

Additional parameters are defined here as all parameters added to those already established for the pinhole camera model as described for a digital camera in section 3.1. This means lens distortion parameters are seen as additional parameters in the context of this project.

The only additional parameters used for the final camera model here are those describing the radial distortion and a “drifting” centre for the radial distortion. The next section discusses this in more detail.

3.3 Lens Distortion Model

Tangential Distortion

The effect of tangential distortion was already noted right after World War II, as cited by Clark & Fryer (1998) and is caused mainly by the imperfect centring of lens components. Tangential distortion, even though not always negligible, is usually an order of magnitude smaller than the effect of radial distortion. Some distortion models simply ignore the tangential effect (Tsai, 1987).

Including the parameters of tangential distortion in the overall camera model adds complexity and consequently processing time to the optimisation routines used for determining lens distortion (section 4.1.3). In order to not completely ignore tangential effects, the centre of radial distortion as described in the next section is allowed to drift or move freely on the image plane separately from the principal point. Stein (1997) claims this is good approximation for the tangential distortion.

Radial Distortion

One of the main deviations from the pinhole model is caused by radial distortion in camera lenses. Radial distortion is caused by imperfect lens curvature due to flawed manufacturing. Even though the manufacturing process usually achieves near-perfect radial symmetry in a lens, the concave profile of the lens is not as easy to

manufacture. For a perfect lens all rays entering it parallel to the optical axis (see Figure 3-1) should intersect perfectly at the focal point lying on this axis. Radial distortion will cause rays to intersect at different points, either further away or nearer to the focal point, causing one of two types of radially symmetric distortions. The two types of radial distortion are shown in Figure 3-4, namely pincushion and barrel distortion. The cross in each sketch indicates the radial centre. Pincushion distortion causes the straight edges of a rectangle symmetrically positioned around the radial centre to curve inwards as shown. For barrel distortion, the straight lines curve away from the radial centre.

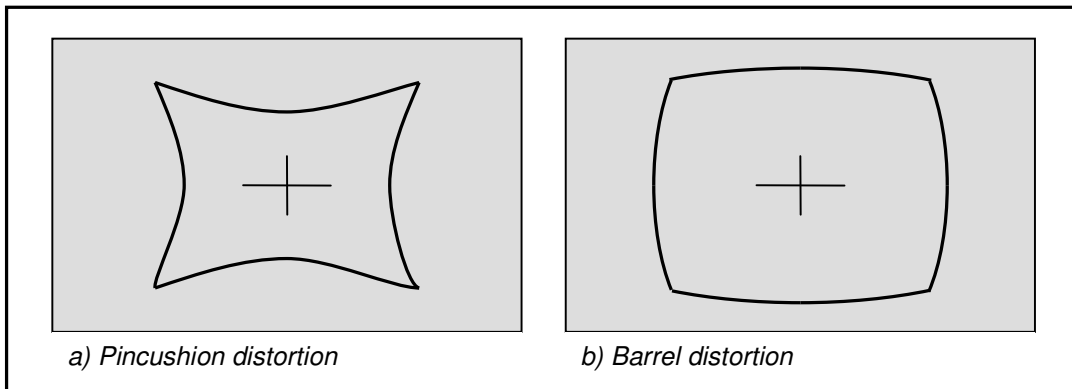


Figure 3-4: Types of radial distortion

Mathematical Model of Radial Distortion

Different mathematical models can be used for radial distort, but they are most commonly described in the form of some polynomial expansion as a function of the distance from the radial centre. The radial distortion model used here was taken from Ma *et al.* (2004:58) and its vector form is shown in Equation 3-13. The model will now be explained using Equation 3-14 to Equation 3-17 and the illustration in Figure 3-5.

$$\mathbf{x}_u = \mathbf{c} + f(r)(\mathbf{x}_d - \mathbf{c}) \quad \text{Equation 3-13}$$

The undistorted image coordinate, \mathbf{x}_u , is computed by adding the coordinates of the centre of radial distortion, \mathbf{c} , to the coordinates of the corrected x- and y-distances. These corrected distances are calculated by multiplying the x- and y-distances from \mathbf{c} to the distorted point, \mathbf{x}_d , by the correction function, $f(r)$ in Equation 3-14.

$$f(r) = 1 + k_1 r + k_2 r^3 \quad \text{Equation 3-14}$$

The distance r is simply the absolute Euclidean distance from the radial centre to the distorted image coordinate and is calculated as shown in Equation 3-15 or Equation 3-16.

$$r = \sqrt{(x_d - x_0)^2 + (y_d - y_0)^2} \quad \text{Equation 3-15}$$

$$r = \|\mathbf{x}_d - \mathbf{c}\| \quad \text{Equation 3-16}$$

The correction function of Equation 3-14 is the most important part of the model, because it mathematically describes the assumed form of the radial distortion for a given lens. The correction function will intuitively either be slightly greater or smaller than one. For barrel distortion, it will be greater than one and for pincushion distortion it will be less than one. Figure 3-5 illustrates the exaggerated correction of a coordinate at a certain radius from the radial centre in an image.

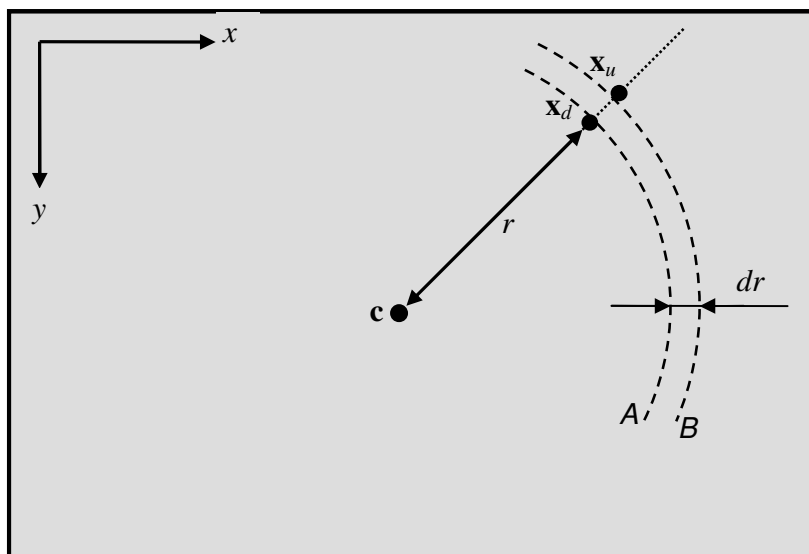


Figure 3-5: Radial distortion explained

In this case the sketch illustrates barrel distortion, because the undistorted coordinate lies further away from the distorted coordinate along the radial line. The radial centre is taken as a free variable and the implementation of this is explained in section 4.1.3. Equation 3-17 shows how the distance dr between the distorted and undistorted coordinate is calculated.

$$dr = r|f(r) - 1| \quad \text{Equation 3-17}$$

The Final Camera Model

The final model combines the pinhole camera model for a digital camera with the additional parameters of lens distortion. The lens distortion model can now include two radial distortion coefficients and a freely moving centre of radial distortion.

Chapter 4 The Measurement Process

Now that the camera model has been established along with some principles of camera calibration, the calibration and triangulation implemented for this project will now be explained. It will be discussed with respect to both the underlying working principles as well as the practical implementation.

4.1 Camera Calibration

4.1.1 The Method

There is freely distributed code available for accurate and established methods such as that of Tsai (1987). Even so, it has been decided that it would be worthwhile to develop calibration code specifically for this project. In this way, a better understanding could be formed of the underlying principles governing accurate calibration. It also allows changes to be made to the camera model and the additional parameters which can aid in developing more accurate calibration methods.

Having discussed the principles of accurate and fast calibration methods (section 2.3), a very simple two-step method has been developed and implemented. In the first step, the camera parameters are approximated using a linear method which ignores non-linear effects such as lens distortion. For this method, a 3D calibration object with known feature coordinates is needed. More about the calibration object is said in section 6.5. The second step introduces non-linear effects of lens distortion with the model described in section 3.3. These parameters are determined through an optimisation function which minimises the back-projection error of the known 3D coordinates using the initial values from the first step.

4.1.2 Step 1: Initialisation of Camera Parameters

If non-linear effects can be ignored, the camera matrix, \mathbf{P} , can be determined using a simple linear method if the image coordinates and their corresponding world coordinates are known. Used here is the DLT method as described by Hartley & Zisserman (2003: 181), but without the minimisation of geometric error. The geometric error minimisation will be mentioned again in the next section. The principles of the steps followed in the code implementation will be briefly discussed here.

Using DLT with Ground Truth

According to Hartley & Zisserman (2003: 179), more than five image coordinates along with their known 3D world coordinates (ground truth) is needed to solve for the camera matrix. There should usually be more coordinates than this for a more robust solution. Hartley & Zisserman suggests the number of point measurements with known world coordinates should exceed the number of unknowns by a factor of five (approximately 30 coordinates or more). This is in agreement with what was mentioned in the literature review. Also, the ground truth coordinates should not all lie in the same plane.

Once the image points of the ground truth coordinates have been extracted as accurately as possible, their coordinates can be accumulated along with the known world coordinates in the form $\mathbf{A}\mathbf{p} = 0$. Matrix \mathbf{A} is $N \times 12$ and contains all the image and world coordinates, while \mathbf{p} is a column vector containing the 12 entries of the camera matrix. If \mathbf{A} only contains eleven rows, the system is solved exactly, but in practice it is almost always over-determined. To solve for an over-determined system such as this, the SVD of \mathbf{A} is calculated and the unit singular vector corresponding to the smallest singular value is taken as the solution of \mathbf{p} (Hartley & Zisserman 2003: 91).

Normalization

For practical implementation of the solution, the linear system first needs to be properly pre-conditioned. This is done by scaling and shifting both the image and world coordinates (Hartley & Zisserman, 2003: 180). For the 3D case, the suggested normalisation is only effective for a relatively compact set of coordinates that lie close to the camera.

After normalisation the DLT algorithm calculates a normalised camera matrix. This matrix has to be denormalised to retrieve the final camera matrix.

4.1.3 Step 2: Refinement of the Camera Parameters

The algorithm suggested by Hartley & Zisserman (2003: 181) includes a non-linear optimisation of the geometric error before the final denormalised camera matrix is retrieved. This step is not included in the calibration process, because it requires intensive optimisation of many variables in the camera matrix. It also does not introduce the more important non-linear effects. The effect of excluding this step has still to be determined and is left for future work.

The next step is to use the values of the camera matrix from the DLT algorithm as initial values for a robust and quickly converging minimisation function. This

function must introduce the non-linear lens distortion into the thus far linear camera model.

Back-projection Error

With the camera matrix and a set of known world-coordinates available, there is an almost intuitive error to minimise: the difference between the calibration-feature coordinates initially extracted from the image and the back-projection of the world-coordinates onto the image plane as shown in Figure 4-1.

To clarify, the camera matrix determined in step one is used to project a known world coordinate onto the camera's image plane using Equation 3-12. The projected coordinate is then compared to the coordinate originally extracted directly from the image. For a perfect calibration and coordinate extraction, the projected coordinate should fall on the exact same position as the extracted coordinate. The Euclidean distance between the two coordinates is called the back-projection error and it can be used as a function-output to be minimised. The camera parameters (linear, non-linear or combinations of both) are given as the function's variable inputs.

With image coordinates given in x-y format ($\mathbf{x}_i = [x_i, y_i]$), the Euclidean distance, d_i , is simply calculated as described in Equation 3-15 or Equation 3-16 (just replace the centre coordinate with the back-projection coordinate). This collection of errors can now be used for minimisation.

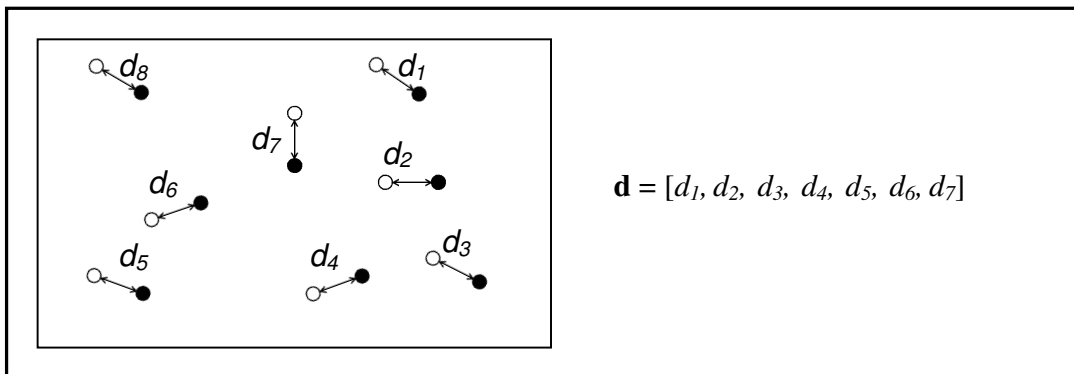


Figure 4-1: Distribution of image and back-projected coordinates

There are different ways in which this error-set can be used to calculate an output for minimisation. Here it has been decided that the mean and standard deviation of the error-set, \mathbf{d} , will be added together and used as the value to be minimised. This has been established through trial and error as the best combination of values. Using the sum of these values gives a low mean value with a higher certainty in the error distribution. Using only the mean usually causes the standard deviation to be slightly higher and vice versa if only the standard deviation is used. Section 8.1 illustrates

how the optimisation function improves the back-projection error distribution with respect to the mean value and the standard deviation. Other values, such as the geometric error (sum of the squared distances) proposed by Hartley & Zisserman, have also been tried, but yield inferior results.

The Minimisation Function

Now that the error to be minimised has been defined, the simple minimisation function is explained using Figure 4-2. The figure illustrates the refinement of the camera parameters for a camera when the lens distortion is introduced into the camera model.

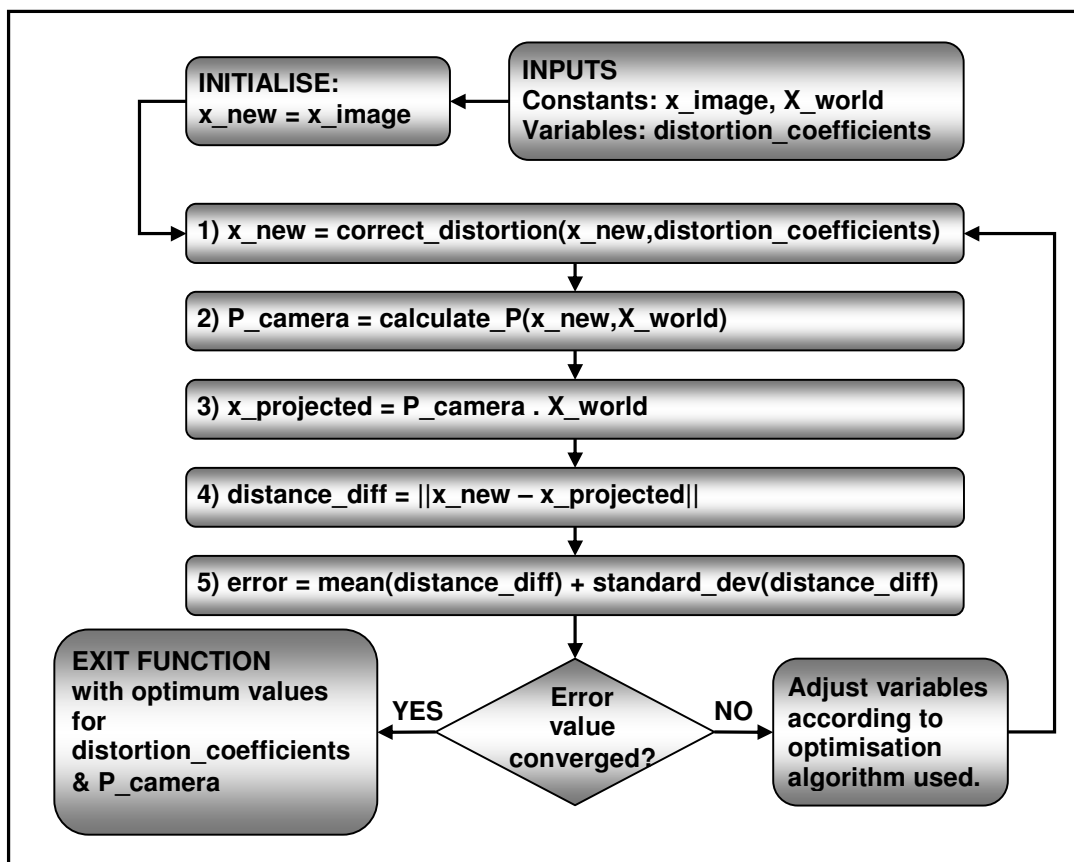


Figure 4-2: Flow-diagram for optimisation function

The known world coordinates and their corresponding image coordinates are given as the constant inputs of the function. The variables to be determined by the optimisation routine are the camera matrix and the distortion coefficients of lens distortion model. Note that the number of distortion coefficients and their meaning is determined by the distortion model used. That is why the collection of distortion

coefficients is simply given a generic name in the figure above. The coefficients are usually initialised to zero, but if the values from some previous calibration can be used it could speed up the optimisation routine.

Once a copy of the image coordinates have been made, step (1) in the minimisation function calculates the new image coordinates according to the current values of the distortion coefficients. For each of the iterations the distortion variables are adjusted. Step (2) uses the known world coordinates and the new image coordinates to calculate the camera matrix. This is done using the DLT algorithm and is in effect the initialisation of the camera parameters as described in section 4.1.2. Step (3) then calculates the projected image coordinates using the corrected image coordinates and the camera matrix.

Keep in mind that the camera matrix has also been calculated using the corrected image coordinates. In this way, the camera matrix is also being optimised along with the distortion coefficients in order to minimise the error. If the camera matrix is constant and the distortion coefficients are determined using only the initial extracted and back-projected image coordinates, the calibration results are less accurate, but the optimisation routine converges faster.

Step (4) calculates the Euclidean distances between the projected and corrected image coordinates. Finally, step (5) calculates the sum of the mean and standard deviation of the error-set of Euclidean distances. If the error value has converged satisfactorily, the routine exits with the optimised distortion coefficients and camera matrix. The optimisation routine used here is that of Broyden, Fletcher, Goldfarb, and Shanno, or BFGS, and it is used as implemented in the SciPy module's optimisation package (section 5.1.1).

Statistical Improvement of the Initial Solution

It can be expected that there will be some coordinates that are not accurately detected. These coordinates will likely cause outliers in the error distribution and have a detrimental effect on the final calibration result. A simple way of addressing this problem is by removing the image coordinates causing outliers in the distribution before calibration. This can only be done after an initial calibration (either with or without non-linear lens distortion effects) has been done to detect the outliers. Assuming that the error is normally distributed, a confidence interval is chosen, outside of which errors are assumed to be outliers. The confidence interval should be chosen carefully in order not to eliminate too many coordinates and so weaken the calibration network. Currently, a confidence interval of 2.5 to 3 times the standard deviation has been found to give good results when minimising the back-projection error.

Note that introducing statistical improvement effectively changes the calibration to an iterative process on a larger scale, because the basic calibration has to be repeated at least twice per camera.

4.2 Triangulation: Measuring Points in Three Dimensions

Once the cameras have been calibrated, it is now possible to find the 3D coordinate of any point for which there is a correspondence in two images. For an exact calibration and image correspondence extraction, the rays formed by the point correspondences in two images should intersect at the 3D coordinate of that point. Unfortunately the process is never exact, causing the rays not to intersect perfectly. There are a number of ways to solve this problem using either linear or non-linear techniques. The DLT method, which is the method of choice in this case, will be covered first, followed by a brief discussion on other available methods.

4.2.1 The DLT Method

Having already used this method in the calibration stages, its simple formulation lends itself towards easy practical implementation. The details of implementation are given by Hartley & Zisserman (2003: 312). The image coordinates and the camera matrix values are once again combined into a matrix, \mathbf{A} , so that it is accumulated into an equation of the form $\mathbf{AX} = 0$. The 3D coordinate, \mathbf{X} , is calculated by composing and solving this equation for each pair of correspondences using SVD. With \mathbf{A} decomposed into \mathbf{U} , \mathbf{S} and \mathbf{V} , \mathbf{X} is the last row of \mathbf{V} .

It is important to note that this linear method is not an optimal estimate for the 3D coordinates, but is still used because of simplicity and speed. There does exist a simple first-order geometric correction, called Sampson-correction, that can be applied to the coordinates for improved results, but only if the error is quite small compared to the measurement (Hartley & Zisserman, 2003: 314).

4.2.2 Other Methods

Another linear method presented by Trucco & Verri (1998: 162) computes the point of minimum distance between two rays. Again, this is not an optimal geometric solution, but is fast and simple to implement.

Hartley & Zisserman (2003: 318) presents an optimal triangulation solution (given the error in the system is Gaussian) that is projectively invariant and non-iterative. It still, however, requires finding the real roots of a 6th degree polynomial.

Ma *et al.* (2004) proposes another optimal solution, but this method is both iterative and requires the minimisation of the back-projection error at every one of the iterations.

These last two methods are considerably more complex and computationally intensive than the linear techniques.

4.3 Process Summary

Knowingly including parts of the measurement process that have not been covered yet, a schematic summary of the final process is given in Figure 4-3. It aims to create an overall view of the process, giving a better idea of where each of the individual steps fit in.

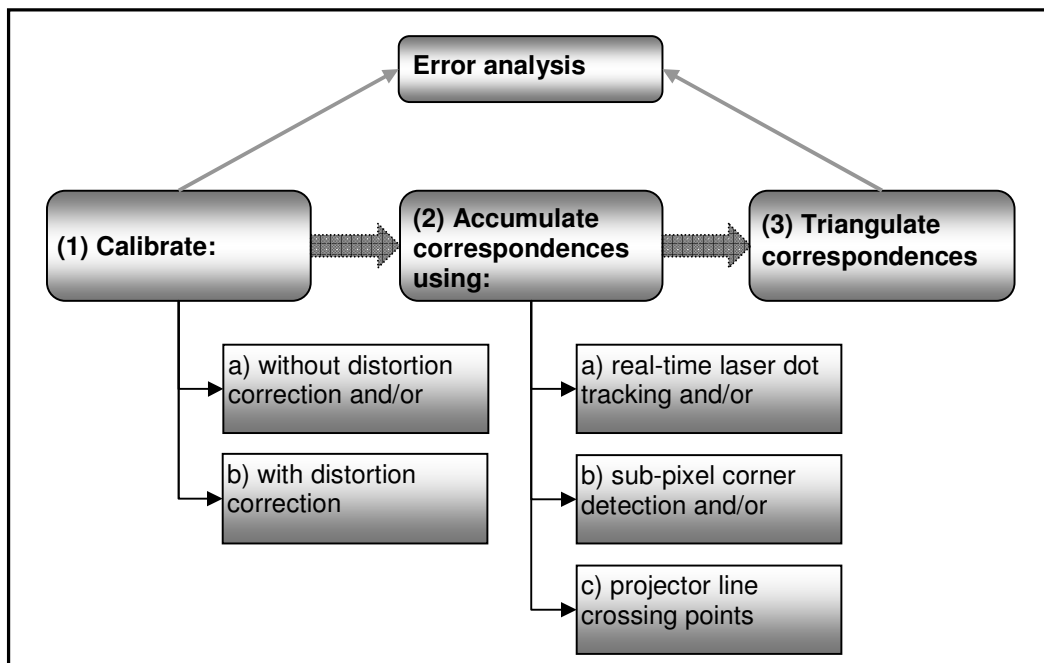


Figure 4-3: Summary of Measurement Process

Steps (1) and (3) in the figure above have been covered with regards to working principles and technical detail. In the code implementation for step (1), the user is given the choice of including or excluding the non-linear effects in the calibration process. Not shown is the option for iterative re-calibration in order to statistically improve the final calibration results. Step (2) is covered in chapter 5, along with the different ways in which point correspondences are determined. The method used for extracting accurate image coordinates for calibration in step (1) is also covered in

chapter 5. Error analysis can be done both directly after calibration as well as after triangulation. The error analysis allows for assessment of the back-projection error, the triangulation error of the calibration object coordinates as well as deviation from a plane after triangulation. This is covered in more detail in chapter 7.

Chapter 5 Image Processing

Image processing plays a significant role in the overall speed of measurement system. Yes, there are certain basic algorithmic complexities inherent to some of the methods (such as calibration) used in the process which also have a deciding role to play in the overall speed and accuracy of the system. However, there are other ways to speed up and automate the measurement process by using the right image processing techniques.

This chapter will cover, both from a practical and technical standpoint, the implementation of these techniques to increase both the system's speed and accuracy.

5.1 Software

Along with low cost, future development of the measurement system is also an important focus of the project. This is why it was decided that only free and open-source software packages distributed under BSD-type licenses would be used. BSD stands for "Berkeley Source Distribution", which is the name of distributions of source code for an operating system developed by the University of California, Berkeley. The license distributed with the code is also now known simply as a BSD-license. Some software licences add small variations to it, but they are still known as BSD-type licences

What is important about this type of license is that it enables software development without the need to pay for licenses and license renewals. It also presents the possibility of commercialising any of the newly developed software.

Open-source software supplies the source of the code in order that it may be altered, improved and added to by others in the software community. With an ever growing community on the web, this means that software packages are improving continually as it is freely distributed amongst users. Packages that fall into this category include the Python scripting language the Intel® OpenCV software package discussed in sections 5.1.1 and 5.1.2 respectively. The remaining sections cover the other important software packages and software related issues.

5.1.1 The Python Scripting Language

As the main programming language, the Python scripting environment is used. It has been established as a very capable programming environment in which to do image processing.

Advantages of Python

As far as execution speed and development time are concerned, it is comparable and in some cases surpasses that of the alternative Matlab® environment. The next important advantage is that the Python license allows free use of the Python language, including many useful external modules, some of which are mentioned in the following section. The license also allows commercialisation of any software developed using Python.

Compared to compiler languages such as C++, Python can reduce development time because it functions as a command-line interpreter. This means that a Python command can be typed in the command-line and executed immediately without first linking and compiling, much the same as the Matlab® environment. Python is designed to emphasise the readability of the code as well as for cross-platform compatibility. This includes compatibility with Windows, Unix-like systems, Macintosh and even the Java and .NET framework.

Python Modules

Python modules are collections of functions and classes that can be imported into Python to greatly increase its functionality. A module is usually focused on a specific software application, such as image processing or vector algebra. Many of these modules are freely distributed under a BSD-type license. Some of the most important modules used in the scope of this thesis include:

- NumPy: for creation and manipulation of multi-dimensional arrays and for linear algebra
- SciPy: for multi-variable optimisation routines, statistical analysis and image processing
- matplotlib: for plotting and image display
- OpenCV: for real-time image processing and display (section 5.1.2)
- MayaVi plug-in: allows commands and data to be sent from Python to the MayaVi 3D visualisation package (section 5.1.3)

5.1.2 The OpenCV Software Package

OpenCV stands for Open Source Computer Vision Library (<http://www.intel.com/technology/computing/opencv/>). It is a collection of C functions and a few C++ classes that implement many commonly used and popular image processing and computer vision algorithms. The package is created and maintained by Intel® and can be used and redistributed under the conditions of their BSD-type license.

Even though written in C and C++, the package supplies bindings to Python. This allows the functions to be called and used from Python with nearly negligible loss in processing time when compared to C++. More about this is said in section 5.1.4.

Images can be loaded as single or multi-channel arrays, each entry containing an intensity or colour value for a given pixel in the image. Many basic array operations can then be done on the image data, such as per-element division, multiplication and scaling to name a few. More advanced processing includes Canny edge detection, morphological filtering, corner detection, Hough line detection and ellipse fitting.

These algorithms were developed with rapid processing in mind. If available and activated, OpenCV can interface with the Intel® Performance Primitives (IPP's), a set of libraries that is optimized for specific processes on an Intel® processor. This can be used to significantly speed up the image processing functions. The IPP's are not implemented in the project, but the current processing speed is sufficient to demonstrate the rapid measurement process.

5.1.3 Data Visualisation

The data that need to be visualised can be placed in two groups according to the type of software package needed to display it: 2D data and 3D data. The 2D data refers to image display and plotting on an x-y axis. The 3D data mainly consists of triangulated image coordinates or 3D meshes representing an image for which the intensity values have been placed on the third axis.

For image display, the OpenCV package offers a simple GUI class that is useful for displaying images real-time when using the digital video cameras (section 6.1). It is not very well suited when images need to be resized interactively or if plots need to be drawn or superimposed on an image.

The Python module Matplotlib has this added functionality with commands that are very much like that of the Matlab® environment. It allows plotted data or displayed images to be interactively resized, panned or moved. As in Matlab®, it is capable of multiple axes in one window, title and axis labels, legends and many other similar functions. Images, plots or combinations of both can then be saved in PNG format directly from the plotting window.

For the display of 3D data-points, Matplotlib has some very basic capabilities, but these are not sufficient for the large data-sets and more advanced visualisations needed for this project. For more elaborate and efficient 3D visualisation the MayaVi data visualiser is used. The MayaVi visualisation window can be accessed from Python through a plug-in: a set of function calls that communicate with MayaVi from Python. This allows use of all MayaVi commands as well as the display window to be called from Python. A script with all the preset commands can be written in Python to load the needed data and to display it using the chosen modules, filters and settings available in MayaVi.

For display of a set of 3D coordinates, MayaVi a couple of different formats. The one used here is the vtk format. Before display is possible it is first needed to convert the 3D coordinate array in Python into a vtk file for Mayavi. The Python module used to do this is called pyvtk and is freely available on the web.

5.1.4 C++ Code in Python: the Wrapping Principle

Although Python is a highly readable programming language that enables fast software development, it is not nearly as efficient at complex iterative processes as C++ or Fortran.

Having this in mind, certain software tools have been developed in order to generate an interface between Python and code written in a language such as C++. This process is called “wrapping” and enables optimised functions written in another programming language to be called from Python with a negligible loss in processing time. In this way, the development speed of Python can still be utilised without a great loss in processing time for specific computationally expensive functions. Some of the main integration packages for interface generation include SWIG, Weave, ctypes (now a standard library distributed with Python), Pyrex and Boost.Python. A summary and comparison for these methods and links to their websites can be found on the web (<http://www.google.com/notebook/public/00116375172106219610/BDRWYlgoQneKc0cYi>). These packages are mostly for C/C++ integration with Python. Packages such as f2py are used for integration with Fortran. For interfacing with C++ code, the Boost.Python libraries are used in this project (<http://www.boost.org/libs/python/doc/index.html>).

The interfacing needed in this project is for communication with the digital video cameras that operate using product-specific C++ functions. Using Boost.Python and an open-source mingw compiler environment, only the most needed camera functions are wrapped for Python. These functions are compiled in a Python file that can be loaded as a standard Python module. The next section has more detail on the functionality and wrapping of the camera software.

5.1.5 Digital Video Camera Software

Section 6.1 discusses the hardware specifications and abilities of the cameras. Almost all of this functionality is linked with and controlled by the software supplied with the cameras. This software includes complete working examples, along with its C++ source code, demonstrating the correct use of the camera functions. It also contains a stand-alone executable from which the camera output can be viewed and properties can be changed.

Some of the more important functions are wrapped to control the following camera properties: colour processing, external triggering, frame-rate, brightness,

shutter-speed and gain. As one of the pre-processing steps of calibration and image capture, a simple interface was written in Python to change these settings while watching the real-time video-feed from two cameras simultaneously. The display of the video-feed and the user interface control was implemented using OpenCV.

The most important and incidentally most challenging procedure that was wrapped is that of frame-grabbing. The camera's image-buffer data must be passed to the OpenCV image structure used in Python. The OpenCV structure in Python is itself wrapped with SWIG and needs to be exposed properly before any pointers can be assigned to it in Boost.Python.

5.2 Automation

In order to achieve rapid object measurement, as much of the measurement process as possible has to be automated. The main steps in this process are currently the pre-processing steps needed for calibration of the cameras and the accumulation of correspondences. These two steps will be briefly discussed and the implementation will then be covered in detail in sections 5.3 and 5.4.

5.2.1 Pre-processing for Calibration

In section 4.1 some of the different methods that can be used in the calibration of a stereo vision system have been mentioned and discussed. In all of these methods some knowledge about the scene is needed. Even in the self-calibration case it is necessary to have enough well-matched correspondences that are distributed across different depths and planes in the scene. Whatever the case, the task of automating the process must utilise all pre-determined scene knowledge to keep user interaction to a minimum. The known geometries on calibration patterns, such as squares or circles, can be automatically detected and sorted.

If an object with easily identifiable features and known feature coordinates is used, the calibration algorithms can be relatively simple. Matching the image coordinates with the known world coordinates, however, is not always a trivial problem. Neither is finding the correspondences of these features in a stereo image pair. These problems must be addressed by automated pre-processing steps. The OpenCV package described in section 5.1.2, for instance, contains a function for automatic detection of all corners in a flat checker-grid pattern. This method will not be used, however, because of the difficulty involved in determining the calibration grid's accuracy.

For reasons mentioned in sections 6.5, a 3D grid was designed and manufactured. A whole new automation process had to be developed and coded for

this grid, with the aim of speeding up and increasing the accuracy of the calibration process.

5.2.2 Point Correspondences in Two Images

The matching process dictates to a large extent at which speed a full metric measurement can be done, because this is usually computationally the most intensive calculation after camera calibration. In general, the methods using some form of structured or active light source are the fastest means of enforcing multiple correspondences in a pair of images. They also have the potential for sub-pixel accuracy and can eliminate any possible erroneous matches, which is important when accurate measurement is needed.

Using structured or active light sources, such as projecting known patterns on an object, mainly aids in the programming of image processing when finding correspondences. Making use of established algorithms and methods such as edge detection and thresholding, the projected patterns overcome the need for easily identifiable features on an object. It also enables the measurement of smoothly contoured surfaces that would otherwise be optically uncooperative.

5.3 Automated Detection of the Calibration Grid

In section 6.5.3 the accurate measurement of the calibration object is discussed. This measurement is done with a CMM to indirectly determine the 3D coordinates of the square's corners on the calibration grid. The image processing methods that were developed attempt to conceptually imitate the 3D CMM measurement method and to apply it to a 2D image. Before this can be done, however, all the square objects have to be identified in the image. This is discussed in section 5.3.2 after the assumptions that were used for the code implementation are covered in section 5.3.1. Following the intermediate steps in section 5.3.3, the sub-pixel approximation of the square corners are finally discussed in section 5.3.4.

5.3.1 Assumptions

In order to simplify the code implementation for the whole process the following assumptions are made:

- (1) The calibration grid's x-axis forms no more than a 40 degree angle offset with respect to the camera's image plane. This is illustrated in Figure 5-1.
- (2) All the squares are visible in both images.
- (3) The squares are well contrasted with the dark background.
- (4) The squares cover a large area of the image.

- (5) The squares are positioned more or less horizontally with respect to the x-axis of the image plane.
- (6) The grid is positioned so that its image is more or less symmetrical in a stereo pair.

Different assumptions relate to different sections of the process's implementation, differing in their effect on various aspects of the measurement process. These assumptions will be referenced from the separately discussed code segments according to the number in the list above.

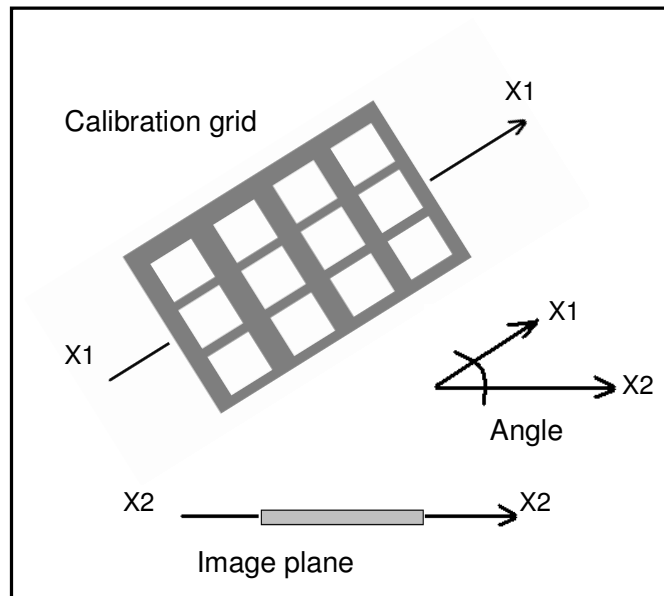


Figure 5-1: Top view of calibration grid and image plane

5.3.2 Finding All Squares

The first step in automating the detection and correspondence matching of the grid corners is to find the approximate positions of all the square surfaces in the images.

Figure 5-2 shows the typical camera views of the calibration grid as it would be positioned for calibration. Assumptions (1) and (3) are relevant in implementing the first main step. Assumption (3) mainly affects the quality of initial edge detection which the rest of the code relies on for effective square detection. Assumption (1) affects the approximation of the polygon (explained later) for a detected square object. If the grid is at too great an angle with the camera, the form of the square becomes elongated and disqualifies it from the polygon fitting. This does not currently stop the square-finding code, but just returns the centre of the object instead of fitting a polygon. The greater consequence of this assumption is found later when the sub-pixel lines have to be fitted to the sides of each square.

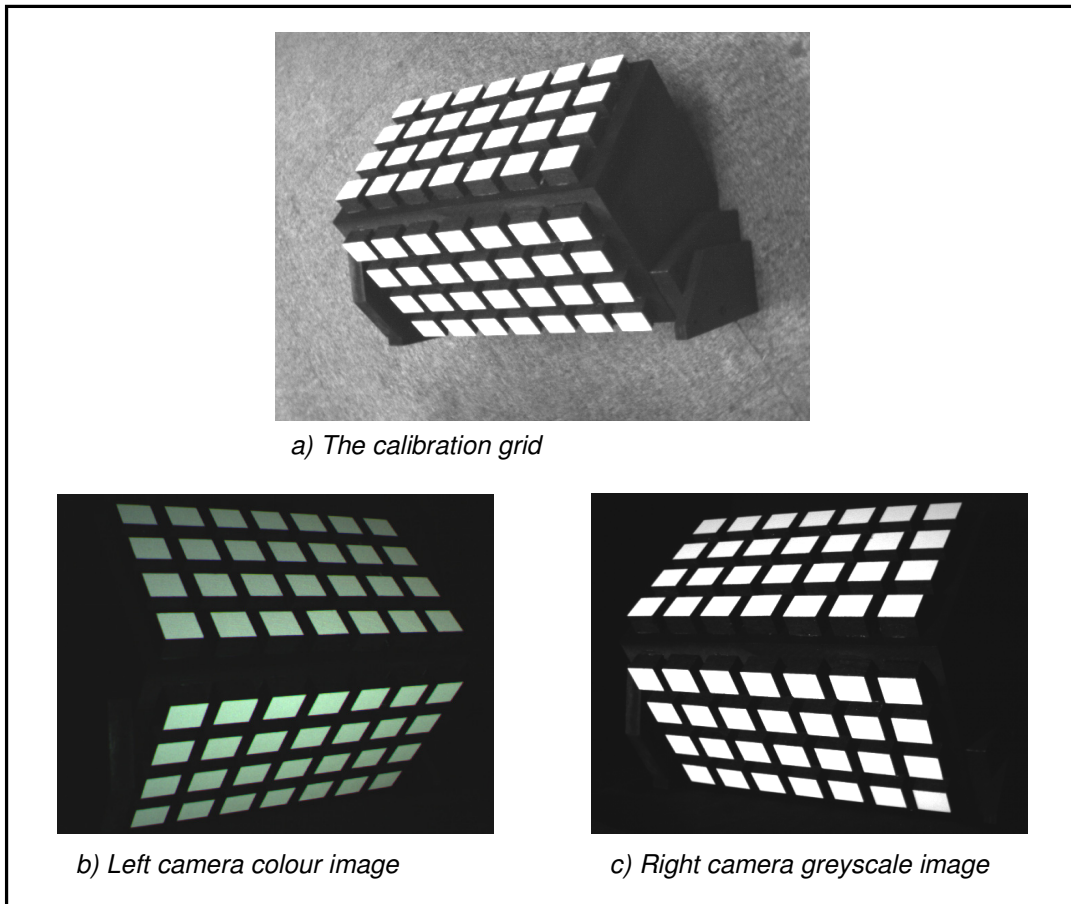


Figure 5-2: Camera views of calibration grid

Code Implementation

Appendix A.1 gives the pseudo-code for initial square-finding process. Figure 5-3 shows the image output after each of the main operations in this part of the process. Note that the image used for this illustration contains artificially added elements to demonstrate the code's ability to remove unwanted objects.

The original image is first filtered for noise using OpenCV's `cvPyr` functions to give Figure 5-3 (a). Applying the `cvCanny` function yields the image in (b). To close any gaps in the edge image, a closing and dilating morphological filter is used to yield image (c). Each closed object in image (c) is then filled and the original edges in (c) subtracted to give (d).

Each of these new objects are then tested for size and rejected if too small compared to the largest object in the image. This yields (e) that only contains square objects.

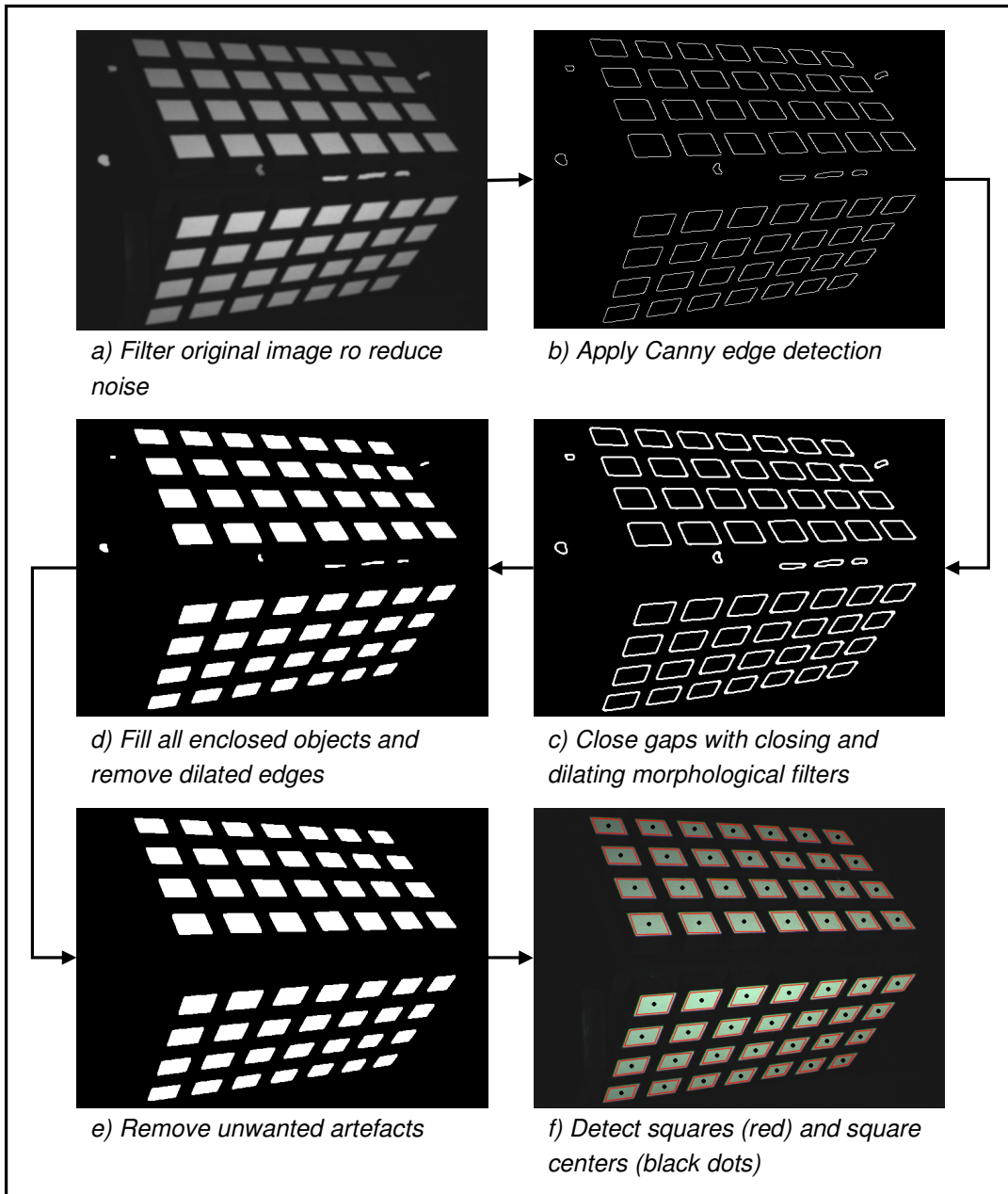


Figure 5-3: Stepwise output of initial square-finding process

Taking the contours in Figure 5-3 (e), the number of sides of each contour is tested to qualify the object for a polygon approximation. For five or six sides the object still qualifies as a square, but only a centre of mass is calculated. The object is still qualified, because some of the corners of the square objects are slightly rounded, causing the cvApproxPoly function to detect extra vertices. If four sides are found

and the polygon is convex and the maximum angle between any two edges is not exceeded, a new polygon is approximated to fit the square-object's sides.

The centres and initial square corners are then returned, as displayed in Figure 5-3 (f). If the image was of good quality, all the square objects will have polygons fitted. If some were not fitted, they are approximated in one of the intermediate functions described next.

5.3.3 Intermediate Steps

Here three less significant steps in the process will be discussed briefly: the sorting of the squares, linking the sorted square centres with available polygons and approximating polygons for any square objects that only have the centre of mass calculated.

The squares need to be sorted so that corner correspondences are immediately matched in another camera image that is in a stereo setup with respect to the first one. The squares are first sorted in rows from left to right and then each row from top to bottom. This sorting is done for each of the sides of the calibration object separately. Figure 5-4 shows the sorting scheme described.

In order to achieve the sorting, the top left or top right corner is first found. The three nearest square centres are then found, which would usually be 2, 3 and 4 in the sketch if the search started at 1. The coordinate with the smallest change in the y-axis would be taken as the next square in the row, unless the end of the row is reached. The change in y-axis position can be used as a valid criterion if assumption (5) is enforced. In the case where the end of a row is reached, the first square of the next row will be used until the last square on the one side of the grid has been reached. Taking the coordinates of the unsorted side, the process is then repeated.

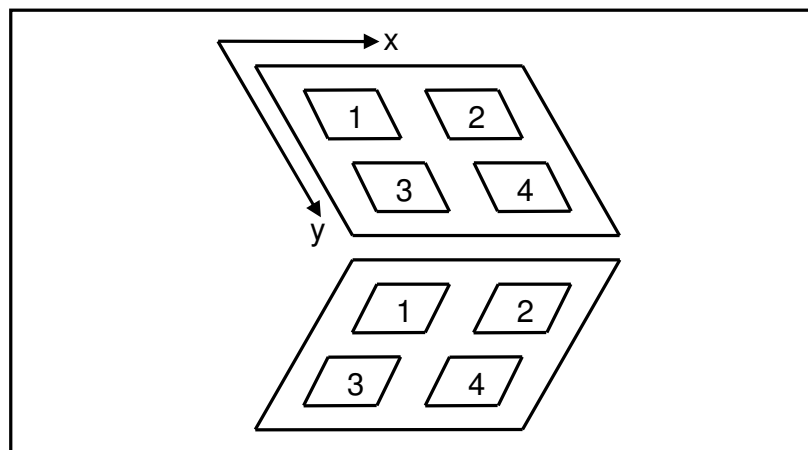


Figure 5-4: Simplified representation of calibration grid

The second step is to link the sorted square centres with the available polygons.

The third intermediate step is to add approximated polygons to any square centres that have not been fitted with one. This is done by calculating which square with a fitted polygon is nearest to a square without one. The nearest available polygon is then translated to the empty position using the offset between the two square centres.

5.3.4 Deriving Sub-pixel Coordinates for Square Corners

While the 3D measurement technique (section 6.5.3) fits four planes to the sides of a wooden block, the 2D method fits a line to each of the four white edges of the block in the image. The 3D method also fits a plane to the white surface of the block and computes the intersection points with the four fitted planes that are perpendicular to the white surface. These intersection points are then the derived corner coordinates. The 2D method calculates the intersection coordinates of the four lines fitted to the sides of the white surface. The pseudo-code for the main function and its most important sub-functions can be seen in appendix A.4.

Figure 5-6 illustrates the stages in the process for deriving sub-pixel corner coordinates while Figure 5-5 shows the kernel used for convolution in order to get the derivative image. It is important to note here that the code for this process was developed before that of 5.4.3. This is why the Sobel operator (kernel) is used only in section 5.4.3, and the simpler one shown in Figure 5-5 is used here and in the final code implementation. The kernel used here is less advanced, but has not been replaced with the Sobel operator, because it is a good illustration of the basic implementation of derivative convolution.

The kernel used here differs from the Sobel operator in two ways: it does not smooth the image for noise and its values do not become smaller as they move further from the zero value in the centre. The last distinction means that this kernel gives the same importance or “weight” to the surrounding pixels used to calculate the derivative for the pixel in the centre of the kernel. The five-element kernel below is the one used in the calibration process, while the one used for the images in Figure 5-6 has seven elements in order to better illustrate the derivative effect.

0.5	0.5	0	-0.5	-0.5
-----	-----	---	------	------

Figure 5-5: Kernel with five elements used for 1D convolution

To get the approximated horizontal derivative image (b) in Figure 5-6, image (a) is convoluted using a seven-element kernel of the same type as the one in Figure 5-5. Image (b) is used to detect edges that are approximately vertical.

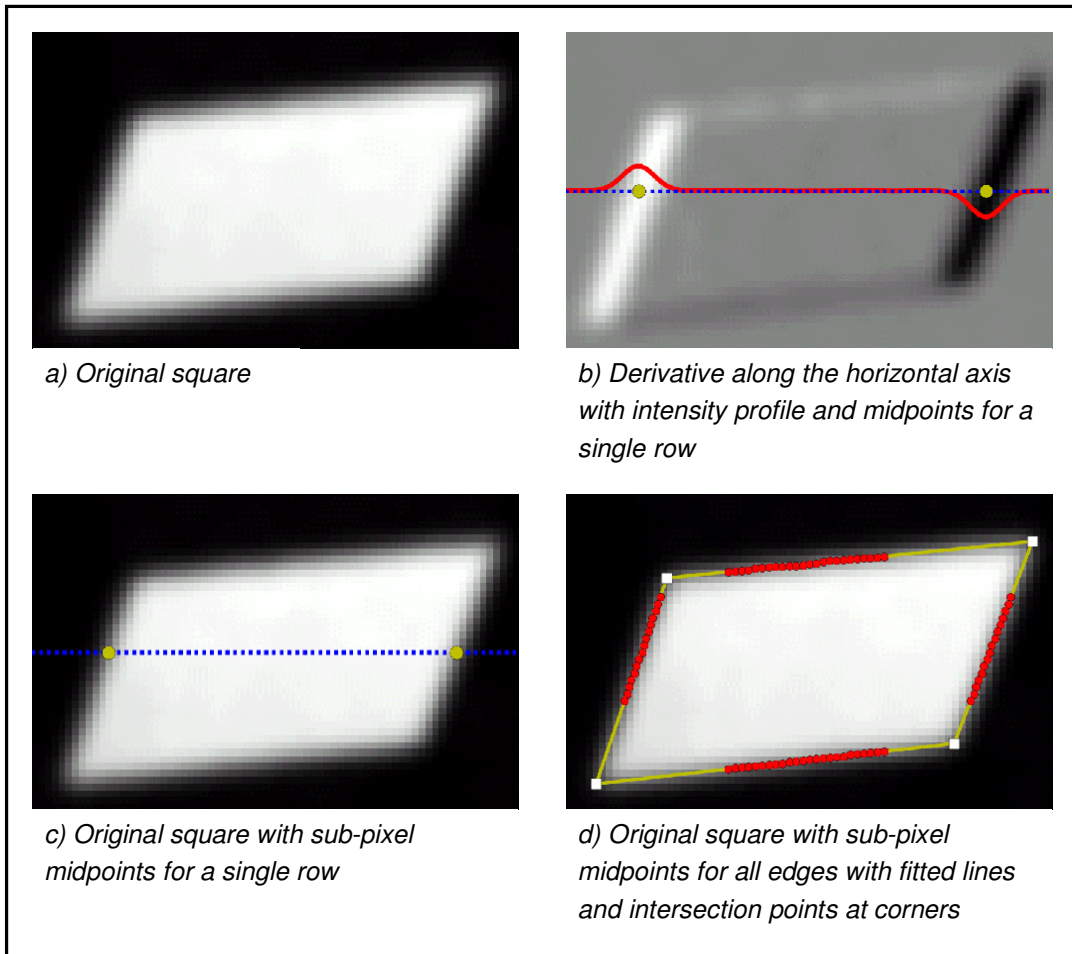


Figure 5-6: Stages in deriving accurate corner locations

To get the horizontal edges, image (a) is convoluted with the transpose of the kernel to yield the vertical derivative approximation.

The derivative image contains positive and negative values, depending on whether the intensities went from low to high (dark to light) or high to low. For sudden intensity changes that would indicate an edge feature, the derivative would yield positive or negative spikes. This is shown in image (b), where the white strip on the left indicates a sudden intensity change from dark to light in the original image. The black strip on the right indicates the light to dark transition. A better visual illustration of the derivative effect can be seen in Figure 5-7 (a) and (b). These two images are just different visual representations of Figure 5-6 (a) and (b). The images in Figure 5-7 use the intensity values as scalar values on a z-axis in order to display the image in 3D.

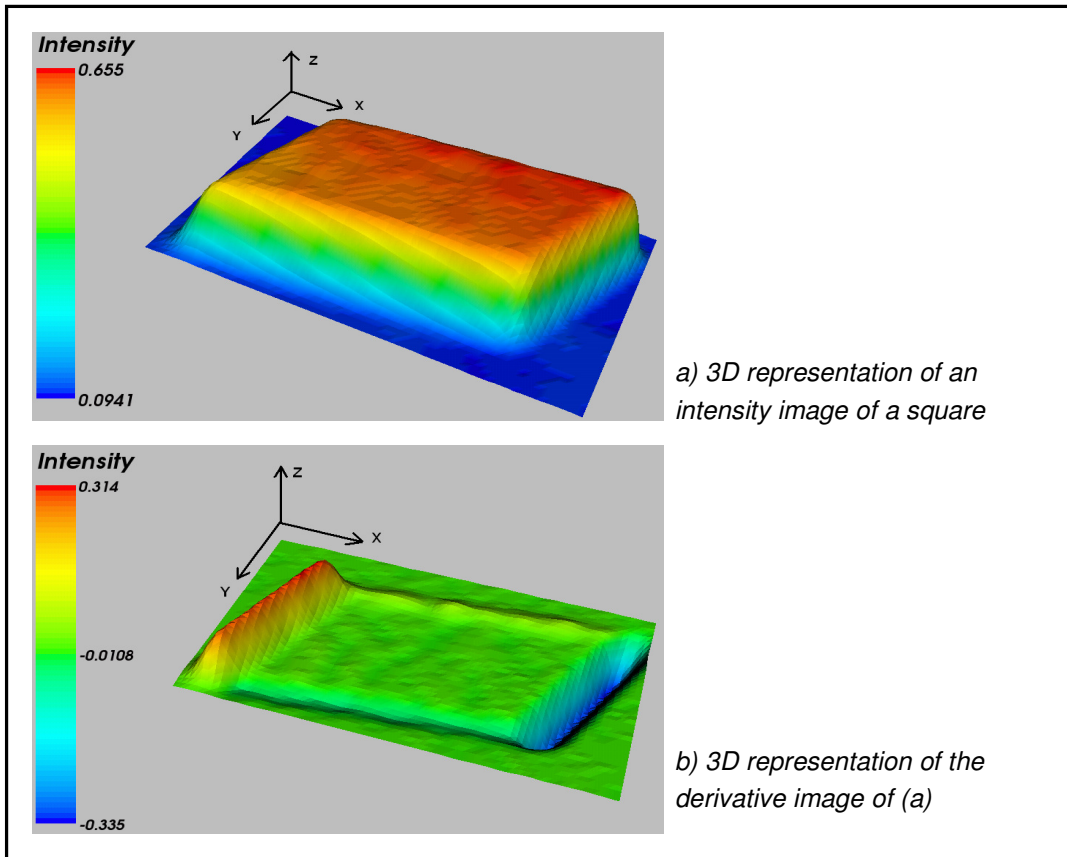


Figure 5-7: 3D representation of intensity images

Starting from the top left corner, the positive x direction (the rows) and the positive y direction (the columns) are indicated with the arrows in both images of Figure 5-7.

To demonstrate how the accurate edge coordinates are calculated, a single row is extracted from image (b). The dotted line in Figure 5-6 (b) and (c) indicates the single row that was extracted from the image. The intensity profile of this row is superimposed on Figure 5-6 (b) plotted as a solid line. This profile with its positive and negative peaks can be better understood when compared to Figure 5-7 (b).

In order to get a sub-pixel position of an edge in the extracted row, the principle of data redundancy is used. It is assumed that the discrete profile formed by the peaks and valleys approximates the form of a parabola for a few points to the left and right of the maximum or minimum values. Taking these points surrounding the maximum or minimum, the coefficients are determined for a parabola that best fits the discrete profile. The location of the parabola's maximum or minimum turning point is then calculated analytically using these coefficients. This location is taken as the new accurate position of the edge in the row. These calculated positions,

indicated by dots, are shown for the extracted line in both image (b) and (c) of Figure 5-6. See appendix A.4 for the code implementation of this step.

This process is repeated for all the relevant rows and columns in the image. Once a number of coordinates are found along each edge of a square image, a line is fitted through each. The intersection points of these lines are then taken as the derived corner coordinates of the square. The results of these last few steps are all shown in image (d) of Figure 5-6.

As mentioned in section 5.3.2, the angle of the grid with respect to the camera's image plane has a considerable effect on the corner calculation. This is because of the current method for detecting edge coordinates. Looking at image (d) in Figure 5-6 it can be seen that only a fraction of the edge is used for the line fitting. This is illustrated more clearly in Figure 5-8 that shows how the edge regions are found.

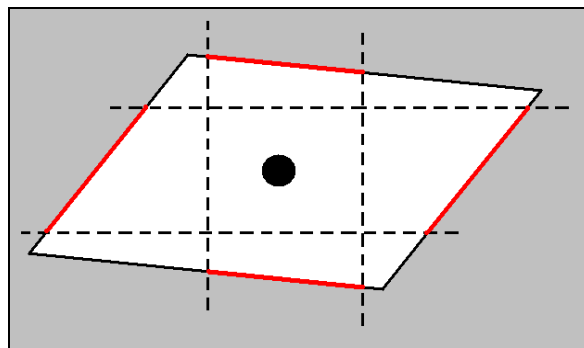


Figure 5-8: Illustration of edge extraction method

The sketch in the figure above shows the typical shape of a square object in an image. The square is projected onto the image plane looking nearly like a parallelogram because of the angle difference between the grid and the image plane. The polygon previously fitted to the square (section 5.3.3) gives an approximated maximum width and height for the square object. Using a fraction of this width and height along with the known approximated centre, the horizontal and vertical search areas are calculated as shown by the vertical and horizontal dashed lines in the figure. The centre is shown by the black circle. Using a small enough fraction of the maximum width and height will ensure that the thicker edge regions bounded by the dotted lines will always lie on the edge of the square object. Unfortunately, the smaller this search area, the less edge information is available, which leads to less accurate line fittings. The fraction of the maximum width and height must therefore be chosen so that the length of the extracted edges is as long possible, but that they will still be guaranteed to fall between the corners of the square. Enforcing assumption (6) ensures that the edges will be found with the same level of accuracy in both images. Assumption (1) ensures that the squares are not elongated to the degree that the search areas have to be made too small for accurate line fitting.

5.4 Rapid Correspondence Matching

Matching corresponding points in a stereo pair of images is one of the most essential parts of stereo-vision. This section describes the three different methods developed and used for correspondence matching in this project. The first method described uses real-time tracking of a moving laser dot. The second and third method uses a DLP projector. All three methods can be classified as active structured light systems (section 2.1.2).

5.4.1 Tracking a Moving Laser Dot

The first and most robust method developed for rapid correspondence matching is achieved by automatically tracking a laser dot moving over an arbitrary surface. This method allows for measurement of objects with complex curvatures. It is currently also the most rapid means of measurement compared to the other two methods. It is capable of accumulating approximately 12 correspondences per second. The other two methods are currently not capable of correspondence matching for complex curvatures as with the first method, but are still used to effectively compare the accuracy and capabilities of the different methods. The last two methods are theoretically capable of complex object measurement, but this is left for future development.

As mentioned earlier, an advantage over the other two implemented methods is that the laser needs no refocusing at different depths.

Motion Detection

Motion detection is achieved by taking two consecutive images from a camera and subtracting them from one another. The image of the difference will then show non-zero intensity values (positive or negative) where movement has taken place and zero intensity values (black) where the scene was stationary. If a high-intensity object then moves in an image sequence, the image of the difference between two consecutive frames will display low negative and high positive intensity values for this movement.

The negative and positive sign of the intensity values in the difference image can be used to determine in which direction the object has moved. While this is not important in the current implementation, it can still be utilized in future work.

Implementation

Figure 5-9 shows an image of a typical laser dot on a flat surface with a square indicating the ROI to be extracted for illustrating the tracking process. Figure 5-10

shows the visual output of the main steps in tracking a moving laser dot such as the one illustrated in Figure 5-9.

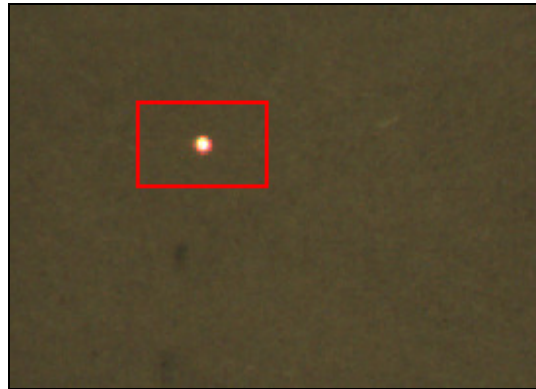


Figure 5-9: Laser dot on flat surface

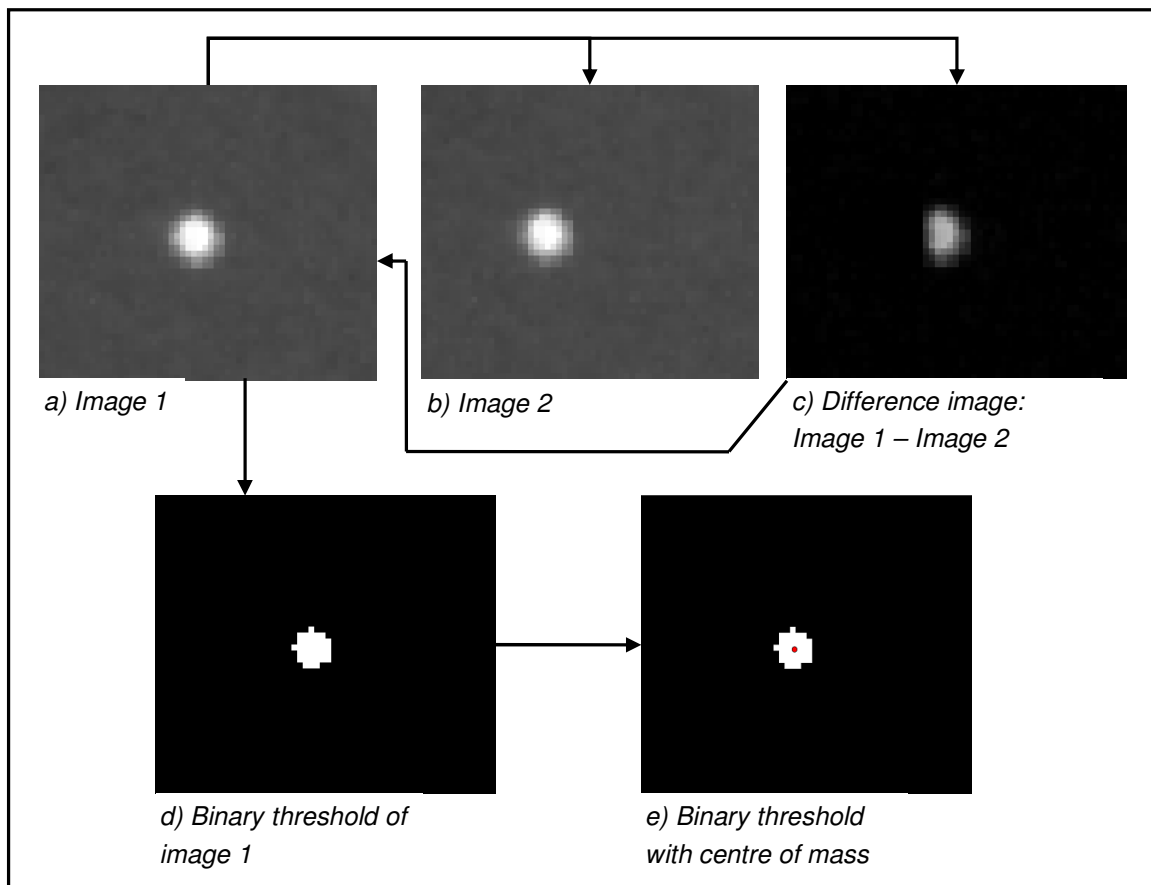


Figure 5-10: Laser-tracking process

The smaller images (a) and (b) in Figure 5-10 were taken from two consecutive full-sized images as the laser moved slightly to the left. The second image is then subtracted from the first image to get the image in (c). If an object has moved, the difference image should give positive and negative values, but only the one or the other is needed in this case. The positive and negative values would correspond to the position of the moving object in the first and second image respectively. The calculation for the image in (c) implemented in OpenCV only gives the positive values indicating the approximate position of object movement in (a). If (c) contains an intensity value above a certain threshold, an ROI is extracted around this intensity value's position in the first full-sized image from which (a) was extracted. This ROI would look like the image in (a) to which a binary threshold is then applied to give the image in (d). Only allowing intensity values above a certain threshold in (c) to qualify laser movement effectively eliminates the detection of moving objects that are darker than the laser. Any brighter object, however, would cause erroneous laser detection. For the code implementation, it is assumed that the laser is the brightest visible moving object in the image. Finally, the centre of mass for (d) is calculated to get a more accurate centre coordinate as indicated in (e).

5.4.2 Corner Detection Using Square Projections

The second method for finding and accumulating correspondences with structured light uses rapid corner detection. The DLP projector casts three white squares on an object (currently only a flat surface has been tested successfully). The corners of these squares are found with sub-pixel accuracy in both cameras using a combination of OpenCV functions. It is currently a semi-automatic process in which the user has to control the projection and image capturing manually. The correspondences are then found automatically.

The pseudo-code for this process is found in appendix A.3. Some important assumptions were made in order to simplify the practical implementation. The first is that only the corner features of the squares will be detected by the OpenCV algorithms. The second is that all the corners will be found in both images.

Finding and Refining Corner Features

Finding the initial corner features is implemented using the OpenCV function *cvFindGoodFeaturesToTrack*. This function uses minimum eigenvalues, calculated using the *cvSobel* function, to detect strong corner features. The integer image coordinates of strong corner features are then returned.

To refine these corners, the *cvFindCornersSubPix* function is used. The function iterates over the image at the discrete corner coordinates to find radial saddle points,

returning their sub-pixel coordinates. For more information on these functions, consult the OpenCV documentation distributed with the package.

Accumulating Corner Correspondences

Using manual input, three squares in a single column are projected onto a flat surface using the highest possible intensity of the projector. This is done to get the best contrast between the white squares and the darker surroundings (see Figure 5-11). If the on-screen display of the current view from the cameras is satisfactory, the user then presses a key to grab an image from each camera for corner feature detection. If the on-screen output indicates that the current set of corners were successfully detected for both images, the user can then project the next set of three squares, slightly offset to the right. The new images with the slightly shifted square projections can then be grabbed again, repeating the correspondence accumulation until the surface has in effect been scanned.

If 12 corners are found in an image grabbed by the camera, these features are sorted in two columns from left to right, with each column's coordinates sorted from top to bottom. It is assumed that a square's corners are more or less horizontally and vertically orientated with respect to the image axis. If 12 coordinates are found in the images from both the first and second camera (assuming the images are of the same stationary scene), these sorted coordinates are stored in list of correspondences. Given the necessary user input, the list of accumulated correspondences is saved to a predefined file-path.

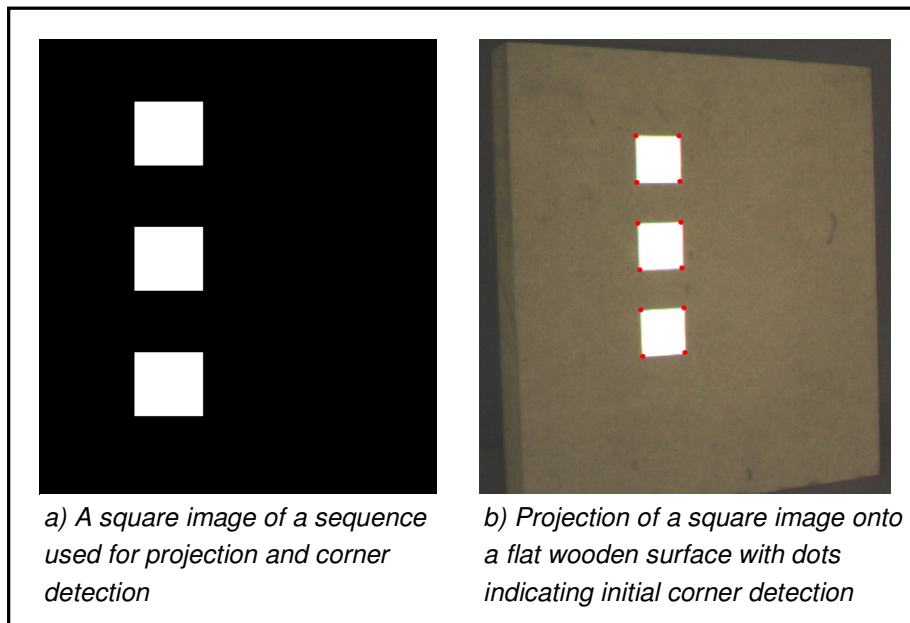


Figure 5-11: Projection of squares for automatic correspondence matching

5.4.3 Projected Line-Crossings

This section describes the process used to create accurate image coordinates using horizontal and vertical light and dark stripes projected onto a stationary object. This process is explained in three parts: the summing of derivative images, calculation of the sub-pixel coordinates and finding the correspondences.

Summing Derivative Images

Figure 5-12 shows the vertically and horizontally projected lines on a flat surface. In order to find the coordinates of the crossing points of those lines, a one-dimensional convolution kernel is used in both the horizontal and vertical direction as in section 5.3.4. The difference in the implementation here is the type of convolution kernel that is used.

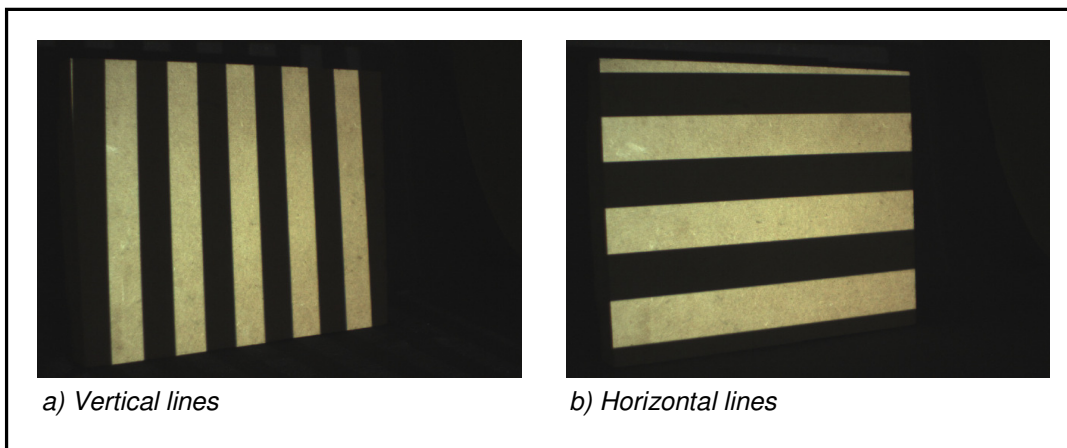


Figure 5-12: Lines projected on flat surface

Convolving the images in Figure 5-12 with a Sobel operator gives the corresponding images in Figure 5-13, scaled to the range of intensity values from 0 to 255.

Figure 5-14 (c) shows the Sobel operator for the horizontal edge detection used to get the derivative image in Figure 5-13 (b). For vertical edges, the transpose of the kernel is used. The Sobel operator consists of two parts: a derivative element and a smoothing element. The smoothing element is added to increase robustness against image noise (Gonzalez and Woods, 2002). The kernel is automatically generated when the `cvSobel` function is used in the OpenCV package (section 5.1.2). Implementation of a 3x3 kernel would have been faster, but the 5x5 kernel uses more image data and subsequently creates more edge data for each crossing point from which to calculate the coordinates. In most cases, more image data means a more robust and accurate calculation of the crossing points. This is briefly discussed in the following section.

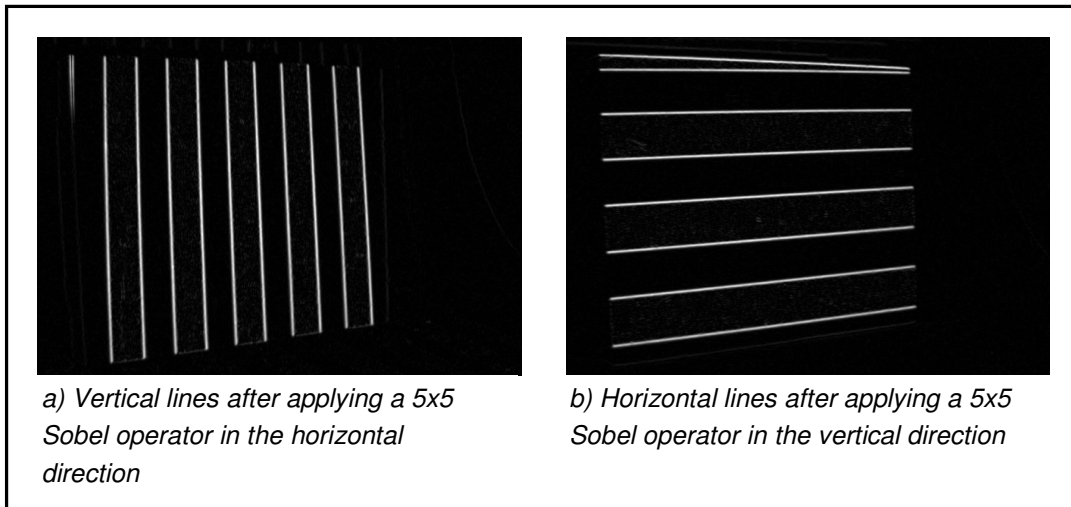


Figure 5-13: Derivative images of lines projected on flat surface.

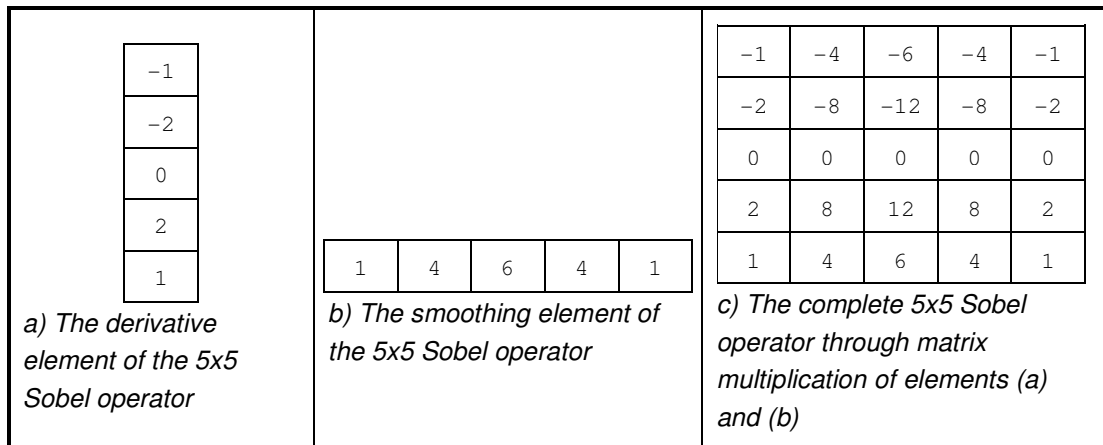


Figure 5-14: 5x5 Sobel operator

Figure 5-15 shows the summation of the two images in Figure 5-13, scaled to the same intensity range. The crossing points can clearly be seen as white spots with higher intensity values than the surrounding lines. The knowledge that these crossing points must have higher intensity values than the rest of the image is used to locate the crossing points.

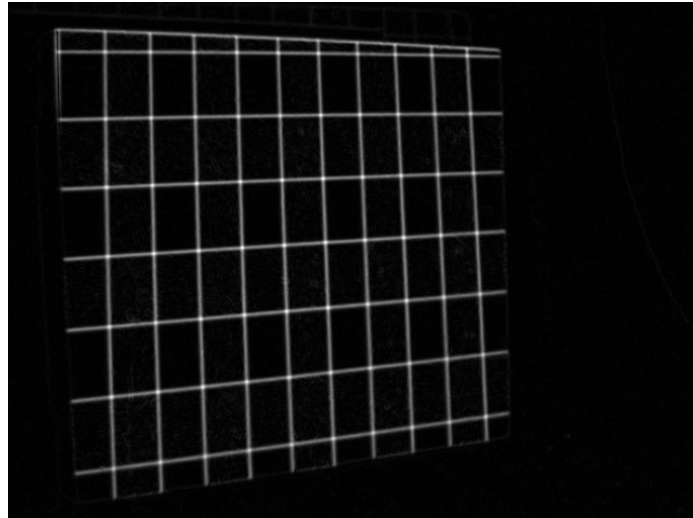


Figure 5-15: Sum of derivative images

Calculating the Crossing Point Coordinates

The implementation of the process finds the position of the current maximum intensity value in a summated image like the one in Figure 5-15. It is assumed that this intensity value will indicate a discrete pixel coordinate of the centre of a crossing point.

Depending on the size of the Sobel operator that was used, an ROI around the maximum intensity coordinate is extracted. This is shown in Figure 5-16 (a) along with a dot indicating the centre of mass calculated for the ROI. The centre-of-mass calculation was done using SciPy's *ndimage* module in Python (see section 5.1.1). The centre of mass is (theoretically at least) a much more accurate floating point coordinate of the crossing point than the discrete pixel coordinate of the maximum intensity value in the ROI. This is because more image data is used. In this case a 12x12 pixel ROI is used instead of the single pixel with the highest intensity value.

Using a smaller kernel would result in less image data describing the parabola-like surface seen in Figure 5-16 (b). This figure better illustrates how the summed derivative images yield a symmetrically looking intensity profile at a crossing point. The peak in the 3D representation corresponds to the pixel coordinate of the maximum intensity value.

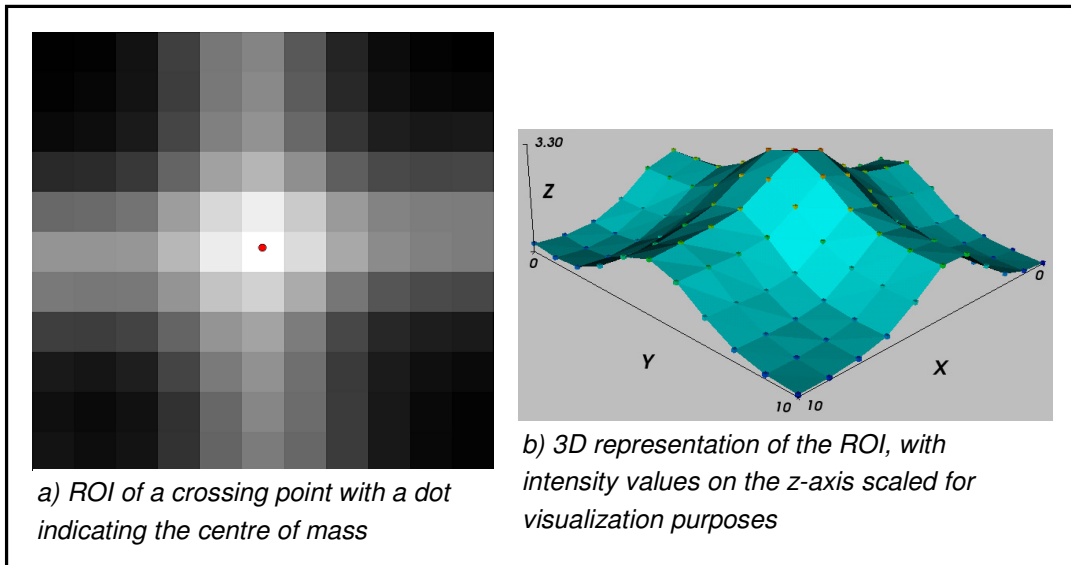


Figure 5-16: ROI around maximum intensity value when 5x5 kernel is used

Finding the Correspondences

Once all the crossing points have been found for the left and the right handed camera, the corresponding points in the two summated images must be determined. A brief explanation of the matching process will be given here.

Assuming that all crossing points are more or less horizontally and vertically distributed, the coordinates for the left camera are sorted from top to bottom and then left to right in separate columns. For each of these sorted points, a corresponding point is searched for in the other image using the fundamental matrix. The fundamental matrix was determined in the calibration process described in section 4.1. With the fundamental matrix known, an epiplolar line can be drawn in the second image for each of the points in the first image.

For an ideal camera model the matching point in the second image has to lie somewhere along this line. Because there are always errors in the model however the matching point will lie very close to the line, but not exactly on it. The matching point is then found by taking all the points that are closer to the line than some predefined distance. The left-most of these points is then assumed to be the correspondence.

Chapter 6 Hardware

6.1 Digital Video Cameras

Two digital video cameras are used in this project, one monochrome, the other colour. Both are Firefly® MV IEEE 1394 cameras distributed by Point Grey Research.

Instead of using two monochrome cameras, a colour camera is introduced to compare the achievable calibration accuracy with that of the monochrome camera. It is also chosen for its potential use in future developments: feature detection by using colour recognition, colour texture mapping of 3D objects and more in-depth studies of the effect of colour images on measurement accuracy.

Some important aspects of the cameras will now be covered.

6.1.1 Camera Properties and Characteristics

Fraser *et al.* (1995) reports that one advantageous feature about CCD (applying also to CMOS) arrays in digital cameras is the “high positional integrity” of the pixel elements, especially for cameras with on-chip A/D conversion. This applies to the FireFly® MV cameras used for this project. Fraser also states that errors attributed to A/D conversion can be rendered “metrically insignificant” through: (1) pixel synchronous A/D conversion and giving “due attention” to (2) camera warm-up and (3) power supply fluctuations.

All three of these requirements have been met using the Firefly® MV cameras. According to the camera specification sheet, the camera pixels have simultaneous integration and readout with a 10-bit on-board A/D converter. For warm-up, the cameras are simply left on for a few minutes before image acquisition. Finally, each camera is powered directly via a single cable attached to a single PC’s 1394 (Firewire) port. The computer already makes compensation for power fluctuations. All the real-time image processing is done by this one computer, equipped with a 3 GHz Intel Pentium 4 processor and 1 GB of DDR RAM.

Digital cameras with CCD image sensors have generally been known to achieve better quality than CMOS sensors. The makers of the CMOS sensors found in the Firefly®, however, claims to achieve CCD image quality. This claim is based on signal-to-noise ratio and low-light sensitivity measurements. It also retains the advantages of size, cost and integration found with CMOS technology when compared to CCD’s. Comparative quality studies have not been done for the cameras, but this is left for future work.

The colour camera has some unique processing characteristics that will now be covered in more detail.

Colour Processing Methods

The raw image data of the colour camera is received as a single layer intensity image. The difference when compared to the monochrome camera is that each pixel has a red, blue or green colour filter in front of it in a configuration called a Bayer-pattern. Knowing the pattern configuration, the intensity values received from the colour camera must somehow be interpolated to create a full three-channel colour image. Each channel then contains the red, blue and green intensity values respectively with each channel being the same size as the original intensity image.

There are four colour processing methods supplied with the camera software. These are: a fast nearest-neighbour interpolation, an advanced nearest-neighbour interpolation, an edge detection method and a computationally intensive rigorous interpolation method. The first three methods can still handle the real-time frame-rate (30 fps) of the cameras, while the last, most accurate method only allows for frame-rates of about 5 fps. The camera documentation claims that the edge-detection method is the most accurate of the three real-time methods and will therefore be used in all of the processing steps.

6.1.2 Lenses

An important consideration when using cameras for photogrammetric type measurements is the effective focal length of the lens. As stated by Fraser *et al.* (1995), long focal length (narrow-angle) lenses have a much less pronounced effect on out-of-plane image deformation (such as radial distortion) than short focal length (wide-angle) lenses. This is because the radial displacement of an image point is a function of the incidence angle of the imaging ray. Wide-angled lenses with greater incidence angles also cause the effect of image-plane deformation (for instance a CMOS chips whose elements do not lie exactly in the same plane) to be greater than for narrow-angle lenses.

The Firefly® cameras are each fitted with a micro-lens that has a 6.37 mm focal length. With its small image sensor size (5.35 mm measured on the diagonal across the chip), this is equivalent to a 51 mm focal length for a standard film camera. For this equivalent focal length, the lens for the digital cameras can be classified as normal, giving very much the same viewing size and angle as the human eye.

The camera can also be fitted with the supplied C-mount lens-holder to accommodate different lenses with longer or shorted focal lengths.

6.1.3 Communication with the Computer

The digital Firefly® cameras do not require separate frame-grabbers as is needed for analogue cameras. After on-chip A/D conversion, the communication is done directly via a single wire to the IEEE 1394 port. For a normal desktop PC, each camera requires only one cable attached to the 1394 port for both data transfer and power.

As mentioned in 5.1.5, the camera comes with all the necessary software and drivers needed to control most of the camera properties. Again, the communication needed for controlling the camera is done via the IEEE 1394 port. Some properties, like the synchronisation, cannot be controlled directly via the computer and needs an external signal (section 6.1.4).

The software also controls the process of accessing the image data from the memory buffer as the images are sent from the camera.

6.1.4 Synchronisation

A critical aspect of multi-camera metrology is that the cameras must be synchronised so that the images of an object can be captured at exactly the same time. This was one of the decisive factors influencing the final choice of cameras for this project.

The Firefly® cameras each allow an external input for a synchronisation signal. The external trigger can be given by either shorting a specified pin to ground or directly driving it from a 3.3 V or 5 V logic output. The latter option was chosen and implemented using an external microcontroller, described in section 6.2. The microcontroller drives both cameras with the same external signal in order to synchronise them.

If the shutter-speed of the camera is too long, the cameras might go out of sync. Even with a short shutter-speed, there is still a chance that one of the cameras might miss a signal. This is dealt with by simply checking the time-stamp for each image currently in the memory buffer. If the time stamps differ, the images are obviously not synchronised. Knowing this, any image processing routine that relies on synchronised images can merely ignore those that are not synchronised.

Even though the cameras are capable of frame-rates of 60 fps at a 640 x 480 resolution, this is reduced to a maximum of 30 fps when an external synchronisation signal is used. Fortunately, this does not impede the performance of the overall system, because the image processing used for target tracking and matching currently limits the achievable speed to about 12 fps.

6.2 External Microcontroller

With many ways to create an external synchronisation signal, the microcontroller used in this project will only be discussed briefly.

The microcontroller used here is Microchip's PIC12F675 mounted on a development board included in the Pickit™ 1 Flash Starter Kit, also from Microchip. The starter kit includes complete source code, written in C, that can be adapted for personal use with the supplied editor and compiler. The development board is connected to the computer via a USB cable to any of the USB ports of the PC. The microcontroller is then powered and programmed using the USB connection.

The existing code was adapted to turn one of the pins of the 8-pin microcontroller on and off to create a square wave switching between 0 and 3.3 V. A capacitor was also added to filter out noise on the rising and falling edges. The frequency of this output signal has been set to approximately 30 Hz in order to reach the maximum achievable synchronised frame rate of the cameras.

6.3 Laser Movement

To ensure a stable and constant laser movement, a motorised two axis rotary platform was built using a Lego Technics set as shown in Figure 6-1. The Lego parts are quick and simple to assemble and modify, ideal for rapid concept development.

The Lego set is supplied with two 9 V DC motors. These are used to achieve the left to right and up and down movement of the laser that is mounted on the platform. Connecting the motors directly to each control axis is not possible, because their rotational speed is too high. A gear-train was added for each motor to severely reduce the rotational speed of each axis. For objects at the expected distance from the laser scanning platform, the scanning speed enables reliable tracking of the laser dot.

Also supplied with the Technics set is a programmable control centre, supplying power to the motors as well as enabling directional control for each motor. The controller can be used manually to control the horizontal and vertical movement of the platform. A short, manually controlled sequence of movements can also be stored in the memory of the controller. The sequence of stored control signals can then be activated and will execute automatically. This can be used to pre-program a sequence of movements that would scan the whole image area covered by the two cameras. A thorough scan of a mould for a plastic bottle is shown in section 8.2.

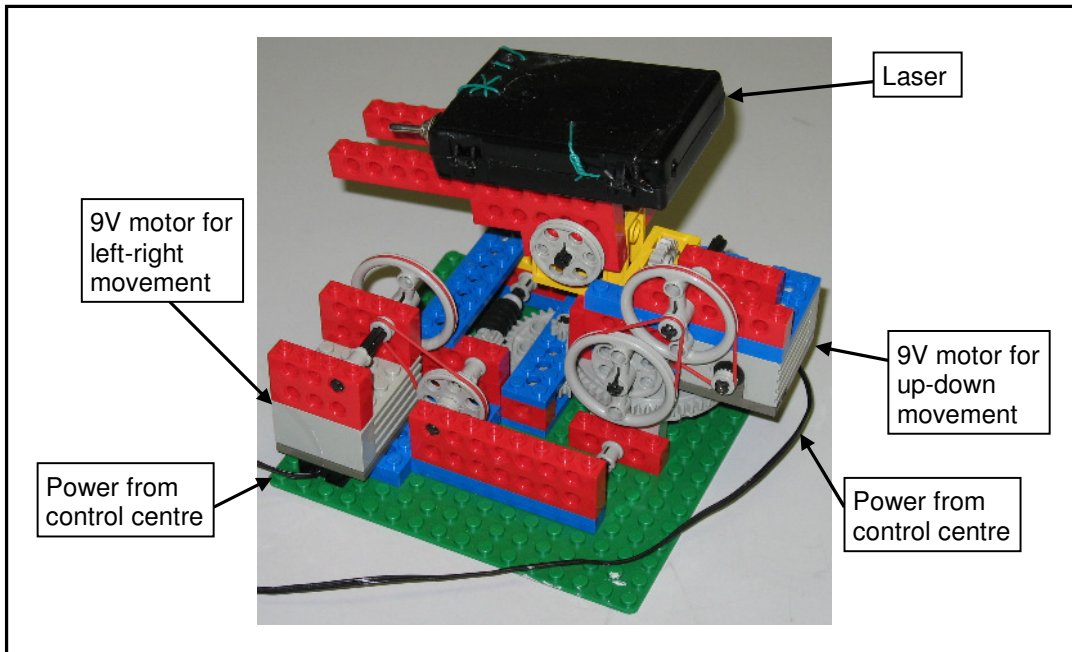


Figure 6-1: Two-axis laser platform

6.4 Projector

As previously mentioned, light projector devices have the advantage of easily displaying a range of patterns on an object. They can also achieve the equivalent of scanning (such as with a laser) over an object without the need for any mobile parts. The main disadvantage is that the projector is only in focus for a specific distance from the projector as apposed to a laser dot or line. Luckily this is not much of a problem if the depth of the object being scanned does not vary too much. Another limiting factor of the projector is its resolution. If, for instance, depth information has to be created for every pixel of the camera's image sensor, the camera's field of view filled by the projector must contain the same or a greater number of projected pixels than the camera. In this way a unique corresponding point can be created for each pixel in the camera.

Currently the projector needs to be controlled manually from another PC in order to switch between the different projected patterns. Ideally, this should be done from the same computer used for the cameras. This is not of great importance for the project, though. The methods using the projector to create correspondences were only added to compare the accuracy of different image coordinate extraction techniques.

Another use of the projector that is very important is illumination. By simply projecting a blank white image from the projector, it creates strong, even illumination for the calibration object. This aids in effective image processing during the calibration process.

6.5 The Calibration Object

It has been decided to use a 3D calibration object. The advantage of using a 3D object instead of a 2D object, such as a planar pattern, is twofold. Firstly, a strong network geometry of well distributed calibration features (see section 2.3.3) can be built into the design. Having the same consistently distributed coordinates during every calibration should aid in getting more consistent calibration and triangulation results. For a planar pattern this is not always so, unless its positions during every calibration remain the same. Secondly, the accuracy of planar patterns are difficult to verify with great precision. This makes a quantitative assessment of the calibration accuracy difficult. With a properly manufactured 3D object, however, very accurate measurement is possible.

The design, manufacture and measurement of the 3D calibration object will be covered next. See Figure 5-2 showing the final calibration grid.

6.5.1 Design

It has previously been established that patterns based on straight lines (rectangles or checkerboard patterns) achieve greater accuracy when used for the extraction of image coordinates. With this in mind, rectangular blocks were chosen with which to create the necessary control points.

In order to benefit from the advantages afforded by coordinate redundancy, multiple blocks are used for each side of the object in evenly spaced rows and columns to further aid in the image processing. To create a strong network geometry, the control points should span the depth of the volume that will be used for measurement. To achieve this, points are distributed on two perpendicular surfaces, with each surface containing a more or less planar set of control points. The control points in this case will be the four corners of each block's contrasted surface. To maximise the calibration volume, the object should also be able to fill as much of the camera view as possible in each camera. This is accomplished by designing the object's width to height ratio to be the same as the image plane's width to height ratio. In this way, when the object is viewed by the camera from the front, the control points can fill the entire frame. Keep in mind that when a stereo pair of images is required, the object will be at an angle with respect to each camera, causing the object to fill a slightly smaller field of view.

Another consideration is the size of the object. It should have more or less the same size as the objects that are to be measured, because its control points should fill the volume that is going to be used for measurement. As mentioned in the literature review, this is done in order to ensure accurate measurement after calibration.

The next criterion is contrast. The rectangular surface of the block has to be well contrasted with the rest of the object structure. The challenge is that only one surface of every block must be contrasted differently and the means used to achieve this must leave the rest of the block visually unaffected. This is addressed in the next section.

6.5.2 Manufacture

To manufacture objects consisting of many sharp corners with the aim of using them as features for measurement is impractical. Sharp features such as corners are easily blunted and cannot be measured directly with accurate touch-probe techniques. Straight-lined features and planar surfaces on the other hand do not have these problems.

For this reason the measurement (discussed in the next section) and the manufacture is focussed on using lines and planar surfaces instead of corner features. As a first experimental calibration object, the blocks are manufactured from chipboard with a white laminated surface coating. The rest of the object structure is made from super-wood.

In order to achieve the necessary contrast, cardboard is glued to the white surface of the chipboard before it is cut into blocks of approximately equal size. The blocks are then spray-painted black while the cardboard is still firmly attached to the surface. After the paint dries, the cardboard is removed and the surfaces cleaned.

The super-wood used for the structure is then screwed together to form the two perpendicular surfaces, with an added foot-piece to keep it upright. The structure is then also spray-painted black. Marking out a grid on the two surfaces, the blocks are glued to it in seven columns and four rows on each surface.

6.5.3 Measurement

To get the necessary accuracy with the specific calibration method that is employed, the world coordinates of the calibration object's control points must be known as accurately as possible. Instead of manufacturing an object within a tight tolerance, the object is rather measured accurately. This is done with a CMM that uses a touch probe to measure one point at a time. The CMM used is a Mitutoyo Bright 710 model with a reported volumetric accuracy of 6 μm .

Deriving the Corner Coordinates

Because a corner feature cannot be measured directly, the strategy is to measure the surfaces of each block. The intersection of the four side planes with the top one is then taken as the derived corner coordinates.

For each of the four black surfaces around the side of a block, four well-distributed point measurements are taken. A plane is then fitted through these points. For the top surface, five points are taken and a plane fitted through them as well. All the calculated corners are accumulated in a text file and later loaded into an array that can be used in the Python scripting environment.

Verifying the Corner Accuracy

Because the four side surfaces of each block are quite rough, some error can be expected in determining the corner coordinates. To verify the repeatability of the corner measurements, the touch-probe measurement is repeated for a single block with a slight offset on the probe each time. In other words, points with a slightly different position are measured on the same side each time. If the sides are perfectly planar, the same corner coordinates should be calculated each time.

The measurement is repeated eight times: enough repetitions to calculate an acceptable standard deviation (assuming systematic errors yielding a Gaussian error distribution). Table 6-1 shows the result of the repeated measurement. For all but one corner, the standard deviation calculated indicates the repeatability is well below 0.1 mm within a 95% certainty interval (three-sigma).

	Corner 1	Corner 2	Corner 3	Corner 4
Three times standard deviation (mm)	0.027	0.067	0.122	0.038

Table 6-1: Certainty of measurement for calibration object corners

Chapter 7 Experimental Setup and Planning

All the implemented methods and individual components have now been covered. These elements are now combined to evaluate the complete measurement system. The combination of the physical components will firstly be discussed, covered by some practical aspects such as illumination and the objects that will be used for measurement. The definition of the type of errors used for evaluating the accuracy will then precede the final section covering the planning of the experiments.

7.1 Positioning the Components

Figure 7-1 shows how the system's components are placed relative to one-another.

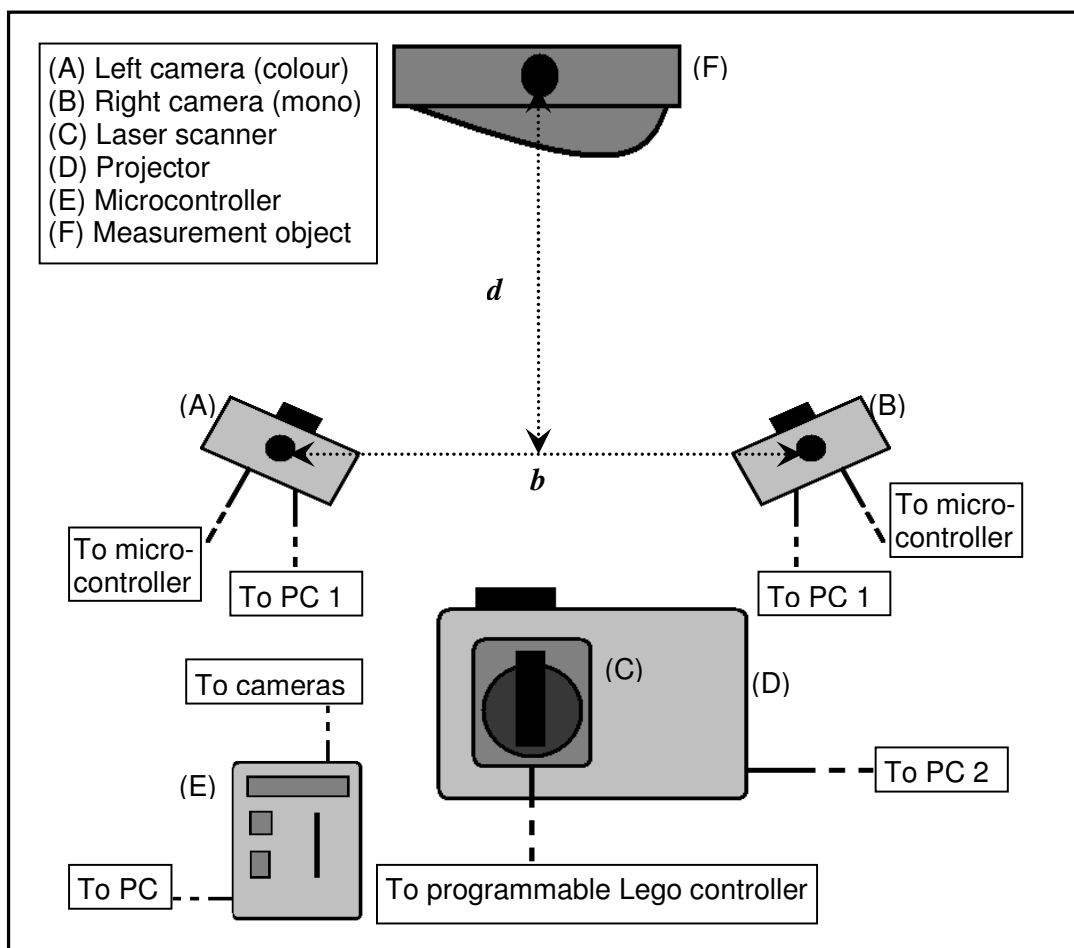


Figure 7-1: Measurement system setup: schematic top-view

During the calibration stage, the measurement object is replaced by the calibration object at more or less the same depth, d . The base-to-depth ratio, b/d , is made as large as possible while still maintaining a practical angle at which correspondences can be detected for non-planar objects. For the standard setup, the ratio is approximately one. As mentioned in the literature review, the accuracy of the system usually improves as the base-to-depth ratio increases, but the extent of the improvement is unknown.

All the parts shown in the figure above are positioned on the same table. The Lego controller is placed off the table so that manual control will not cause any object or camera movement that might affect measurement. The microcontroller (E) can be powered by attaching it to any one of the PC's via the USB cable.

Figure 7-2 shows the actual setup, with the component labels corresponding to those in Figure 7-1. Again, the measurement object, (F), is replaced by the calibration object at approximately the same position during the calibration phase.

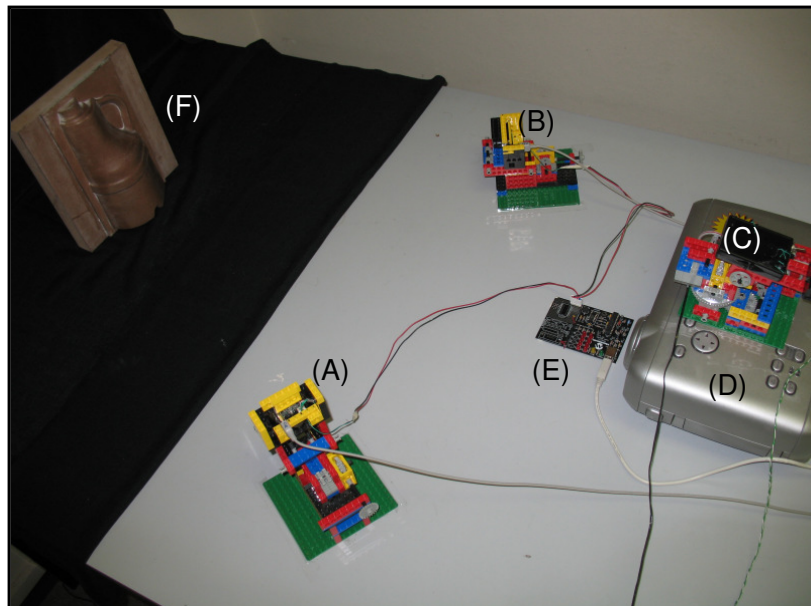


Figure 7-2: Actual measurement system setup

7.2 Illumination

There are commonly known errors introduced into optical measurement systems because of non-ideal lighting conditions. Insufficient or non-uniform lighting makes image-processing more challenging. Over-exposed objects, on the other hand, can cause blooming effects which causes a deterioration in the accuracy of location extraction.

The DLP projector is used to achieve sufficient and uniform illumination over the calibration object during the calibration phase. A simple white image is given to the projector as output, covering all the rectangular blocks on the calibration grid. Following the example of Fraser (1995), the blooming effect is minimised manually by visual inspection and adjustment. Here, however, it is done by controlling the shutter-speed of the cameras via the software interface instead of adjusting the light-source.

Another potential problem, especially during the calibration phase, is high-contrast features in the background that the image processing does not make provision for. This is addressed by covering the background with a black cloth as can be partially seen in Figure 7-2.

7.3 Objects Used for Measurement

Including the calibration object, there is only one other object used for measurement, but its use is twofold. It consists of a thick wooden board with half a wooden profile of a bottle glued to it. This object is shown in Figure 7-2 and is labelled (F). The flat back of the board is used for the test of deviation from planarity (section 7.4.3), while the bottle profile is used only for a qualitative gauge of the measurement system's capability (section 8.3).

As mentioned, the calibration object itself is also measured in the tests. After the initial calibration, the stereo image pair of the calibration object is used to triangulate the coordinates of the extracted corner features. The triangulation result can then be compared with the known coordinates as described in section 7.4.2.

7.4 Definition of Error Measurements

Objects of theoretically any size can be captured on an image if taken from the right distance or if the right lenses are used. From entire mountain ranges to very small surface-mounted electronic components, objects of different sizes can be captured and displayed on the same image plane. In the case of digital cameras this means that the width or height of a pixel element can represent a few microns in one image, but a few meters in another image.

For comparison with other optic measurement systems, some accuracy representation is needed that does not vary with object size. This can be achieved by the back-projection of known world coordinates onto the image plane. This is will be discussed in the next section as the first measure for the system's accuracy. The second type of accuracy evaluation is the metric triangulation error measured using

the calibration grid. Lastly, a means will be presented to evaluate the accuracy by measuring the deviation from a fitted plane.

To summarise, three different ways will be presented to evaluate the system's accuracy. Changes to the system will be made in a systematic way in order to evaluate their effect on the three different error measurements.

7.4.1 Back-projection Error

The function that is minimised to determine the lens distortion coefficients makes use of the back-projection of world-coordinates onto the image plane. This is already discussed in chapter 4 and needs no further explanation here. A few things will still be mentioned about the display and evaluation of back-projection error.

The first means of displaying this error is done using a histogram. The histogram can be visually (or mathematically) evaluated to qualify it as a Gaussian distribution. Visual evaluation and comparisons with other histograms (using the same scale) can also help determine the degree of improvement from one test to the next.

The second visual representation displays the error in the image plane for each back-projected coordinate. This representation is useful for determining whether the errors follow some predictable pattern, such as increasing errors for points further from the radial centre. An illustration of this can be seen in section 8.1.2.

7.4.2 Triangulation Error

The triangulation error is calculated by taking the triangulated results of the grid's corner features and seeing how much they differ from the known coordinates. This is implemented in much the same way as the back-projection error, only in this case for 3D coordinates. The Euclidean distance between each triangulated point and its corresponding known world coordinate is first calculated. As explained for the back-projection error in section 4.1.3, these Euclidean distances, each being a single error measurement, is combined into one error-set. The triangulation error is also displayed in the form of a histogram for statistical evaluation.

7.4.3 Deviation from a Fitted Plane

Both Chi-Fang & Chi-Yang (1999) and Zhang & Huang (2006) use the deviation from a flat surface as an estimate of the noise in the measurement system. The principle is that if a flat surface is reconstructed, any deviation from planarity indicates the basic measurement error that can be expected in the system.

Verifying the Test Objects Surface Flatness

To use a surface for this kind of error measurement, its actual deviation from flatness needs to be known in order to properly interpret the final optical measurement results.

As with the calibration object, the surface of the test object is measured with the touch-probe of the Mitutoyo CMM. A set of 24 well-distributed points are measured across the surface and the minimum to maximum range of the error is calculated as 0.13 mm. The standard deviation is 0.036 mm, predicting a “flatness” of 0.144 mm when four times the standard deviation is used which gives a confidence interval of approximately 98%. This means that if enough points on the plain are measured, the deviation of 98% of the points should lie within a range of 0.144 mm.

Calculating the Deviation

A plane can be described in a 3D coordinate frame according Equation 7-1.

$$aX + bY + cZ + d = 0 \quad \text{Equation 7-1}$$

The coefficients (in this case to be determined by a least-squares fitting) are a , b and c while X , Y and Z are the 3D coordinates.

To acquire the set of 3D coordinates, a flat surface is scanned using one of the techniques described in section 5.4. These correspondences are then triangulated to give a set of 3D coordinates that should theoretically lie on the same plane.

The set of coordinates is stacked together in a matrix \mathbf{A} to get an equation of the form $\mathbf{A}\mathbf{b} = 0$. This equation is solved in a least-squares sense again using the SVD technique described previously to get the vector \mathbf{b} containing the coefficients.

With the coefficients for the ideal plane known, the shortest Euclidean distance of each 3D coordinate to the plane can be calculated using Equation 7-2.

$$D = \frac{aX + bY + cZ + d}{\sqrt{a^2 + b^2 + c^2}} \quad \text{Equation 7-2}$$

The distance of every point to the fitted plane is collected in one data-set and four times the standard deviation (four-sigma) is computed as an indication of the error.

7.5 Planning the Experiments

A number of experiments are conducted to evaluate different aspects of the measurement system. The errors defined in the previous section will be used as the output for each of these tests. The system will be evaluated in three ways. Firstly, a few parameters of the system will be varied to determine their quantitative effect on

the accuracy. Another test during this first evaluation step will be to validate that repeated experiments with a slight displacement of the calibration object give average results representative of the specific case. Secondly, having fixed the parameters from the first step, the different correspondence matching methods will be tested. Keep in mind that the second evaluation only uses a flat surface for measurement. The final evaluation, even though only qualitative, will be made using the practical measurement object mentioned in section 7.3.

The first two evaluations will now be discussed in more detail.

7.5.1 Variable Parameters and Variability

The main influences on the system's accuracy have already been established from the literature review. Only two basic parameters will therefore be chosen as variables to be tested. The first system parameter chosen as a test variable is the base-to-depth ratio. Secondly, the effect of the camera model's complexity will be tested.

Because it is already addressed in the literature review, the first of these three variables has a predictable outcome. It is still deemed an important test, because quantitative values for its effect have not been found in studies that can be directly compared with this project. For much the same reason the effect of the camera model complexity is tested as well: its quantitative effect in the context of this project is unknown.

It is assumed that the effect of the two variable parameters are independent of one-another. An experimental design testing the interdependence of the variables, such as a full-factorial experimental design, is therefore not used. For each variable, the other parameters are held fixed.

For each of the two abovementioned tests, average values will be presented following five consecutive calibrations. For each calibration, the calibration object will be slightly displaced. This is done in order to get results that are more representative of the specific test-case. Some results for five consecutive calibrations will be given for one specific test, also including their standard deviations. If the standard deviations are small enough, the average values are validated as being representative of the specific test-case.

7.5.2 Correspondence Matching

The values for the triangulation accuracy calculated from the calibration cannot be used as a direct evaluation of the measurement accuracy that the system can practically achieve. There are two reasons for this. Firstly, the calibration was optimised specifically for the calibration object coordinates. The final triangulation result can therefore be more favourable for these coordinates than for other arbitrary coordinates. Measuring points in the same volume spanned by the calibration

coordinates should, however, take care of this. Secondly and more importantly, the measurements made with the structured light use other methods for extracting image coordinates than the calibration phase.

The calibration's coordinate extraction technique for the corner features has already been established in the literature review as the most accurate, assuming moderate lens distortion. Being computationally intensive, this method could not be used for the rapid correspondence matching needed for the final measurement phase. The three methods used for measurement will therefore also be tested for accuracy using the planar deviation error as output.

Chapter 8 Experiments and Results

The results of the different system evaluations as discussed in chapter 7 are now presented.

8.1 Variable Parameters and Variability

The effect of the base-to-depth ratio and the complexity of the camera model is now presented. For each test, the other variables are set to their “standard” values, in other words, as they would be for the evaluation in section 8.2. The standard values for each of the variables are: a base-to-depth ratio of one and a camera model including all three of the lens distortion parameters.

Each value presented in the tables for the variable parameters is the average calculated after five consecutive calibrations with a slight displacement in the calibration object each time. The complete data set for all five runs of every experiment is given in appendix B.

Following the three variable parameter tests, results are shown for the consecutive calibrations in order to validate the use of the average values.

8.1.1 Base-to-depth Ratio

The base-to-depth ratio is defined in section 7.1. For this test a single precise value of the ratio cannot be practically established. This is because the calibration object is so close to the cameras that it causes the ratio to vary significantly between the features on the object. The depth value is therefore chosen as the approximate distance to the centre of the object, much the same as the illustration in Figure 7-1. For the different test runs, the calibration object remains in the same position while the cameras are moved further from or nearer to one another across the baseline (the line along which the base distance is measured).

Table 8-1 shows the results of the back-projection error for the two approximate base-to-depth ratios, while Table 8-2 shows the triangulation results.

Even though the 0.5 ratio yields better back-projection results for the colour camera (Table 8-1), this does not mean it will give better triangulation results. After five consecutive runs to get the average values presented in the tables, it is clear that for a greater base-to-depth ratio the triangulation is more accurate.

Base/depth ratio	1	0.5
Colour camera		
Mean (pixels)	0.219	0.214
Std. deviation (pixels)	0.116	0.114
Monochrome camera		
Mean (pixels)	0.225	0.272
Std. deviation (pixels)	0.123	0.138

Table 8-1: Back-projection errors for varying base-to-depth ratios

Base/depth ratio	1	0.5
Mean (mm)	0.157	0.214
Std. deviation (mm)	0.077	0.118
Precision, 95% confidence (mm)	0.388	0.569

Table 8-2: Triangulation errors for varying base-to-depth ratios

8.1.2 Camera Model Complexity

Because the calibration code was specifically developed for this project, the camera model can be changed easily. It has already been mentioned that a more complex model does not necessarily yield more accurate results (section 2.3). To test the effect of increasing model complexity, different combinations of the lens distortion parameters are used in the camera model for each calibration. The first test uses the DLT method directly with no distortion parameters. The first radial distortion coefficient, k_1 , is then introduced, followed by the second, k_2 , and finally the drifting radial centre, c , is also added. Table 8-3 and Table 8-4 give the back-projection and triangulation results of calibration respectively.

The very small difference in the triangulation error between the last two columns of Table 8-4 indicates that the tangential distortion has a much smaller effect on accuracy than the radial distortion. To clarify: when adding the drifting centre to the distortion model, the improvement in accuracy is two orders of magnitude smaller than the improvement gained for adding radial distortion.

When using only one distortion coefficient, the accuracy is still comparably close to the more accurate cases. Using only the linear model, however, yields results that are significantly less accurate, even with the iterative improvement that gets rid of statistical outliers.

Camera model	Pinhole model	k_1	k_1, k_2	k_1, k_2, c
Colour Camera				
Mean (pixels)	0.353	0.223	0.216	0.206
Std. deviation (pixels)	0.191	0.122	0.116	0.114
Monochrome Camera				
Mean (pixels)	0.431	0.248	0.235	0.231
Std. deviation (pixels)	0.237	0.142	0.133	0.129

Table 8-3: Back-projection errors for different camera model complexities

Camera model	Pinhole model	k_1	k_1, k_2	k_1, k_2, c
Mean (pixels)	0.266	0.163	0.156	0.153
Std. deviation (pixels)	0.122	0.079	0.073	0.073
Precision, 95% confidence	0.632	0.400	0.375	0.371

Table 8-4: Triangulation errors for different camera model complexities

Figure 8-1 and Figure 8-2 compares the back-projection errors and triangulation errors for the worst and the best results namely the pinhole case and the complete camera model case respectively. The back-projection is only shown for the monochrome camera. The visualization (a) of back-projection error on the image plane shows the direction and magnitude (30 times the actual pixel error) with the red lines on the extracted image coordinates.

Figure 8-1 (a) shows a systematic pattern indicative of lens distortion. This is to be expected since no lens distortion has been compensated for. It is also clear from a simple visual comparison that the error for image (a) is much greater than for image (b). For image (b), there does not seem to be a systematic pattern, indicating that the distortion correction did not simply lessen the distortion effect, but also removed the systematic increase of its effect closer to the edges of the image. The missing coordinates in the two images of Figure 8-1 are those that were identified as outliers in the iterative statistical improvement of the calibration result (section 4.1.3).

In image (a) there seems to be a tendency for outliers to be detected near the edges of the image, which is not the case for image (b).

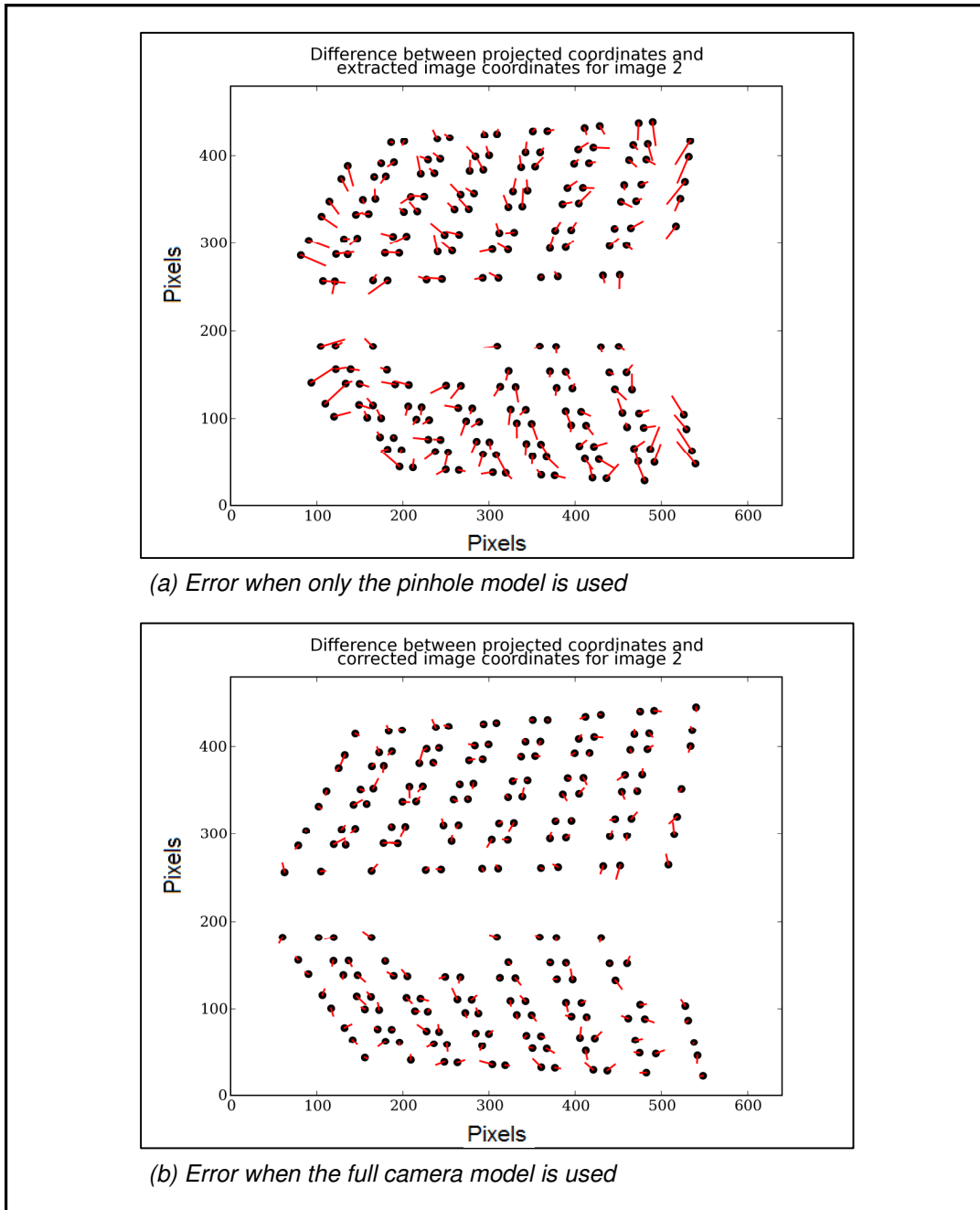


Figure 8-1: Back-projection errors for different camera models

Figure 8-2 clearly shows how the error distribution is improved from the pinhole model to the full camera model when comparing the final triangulation results.

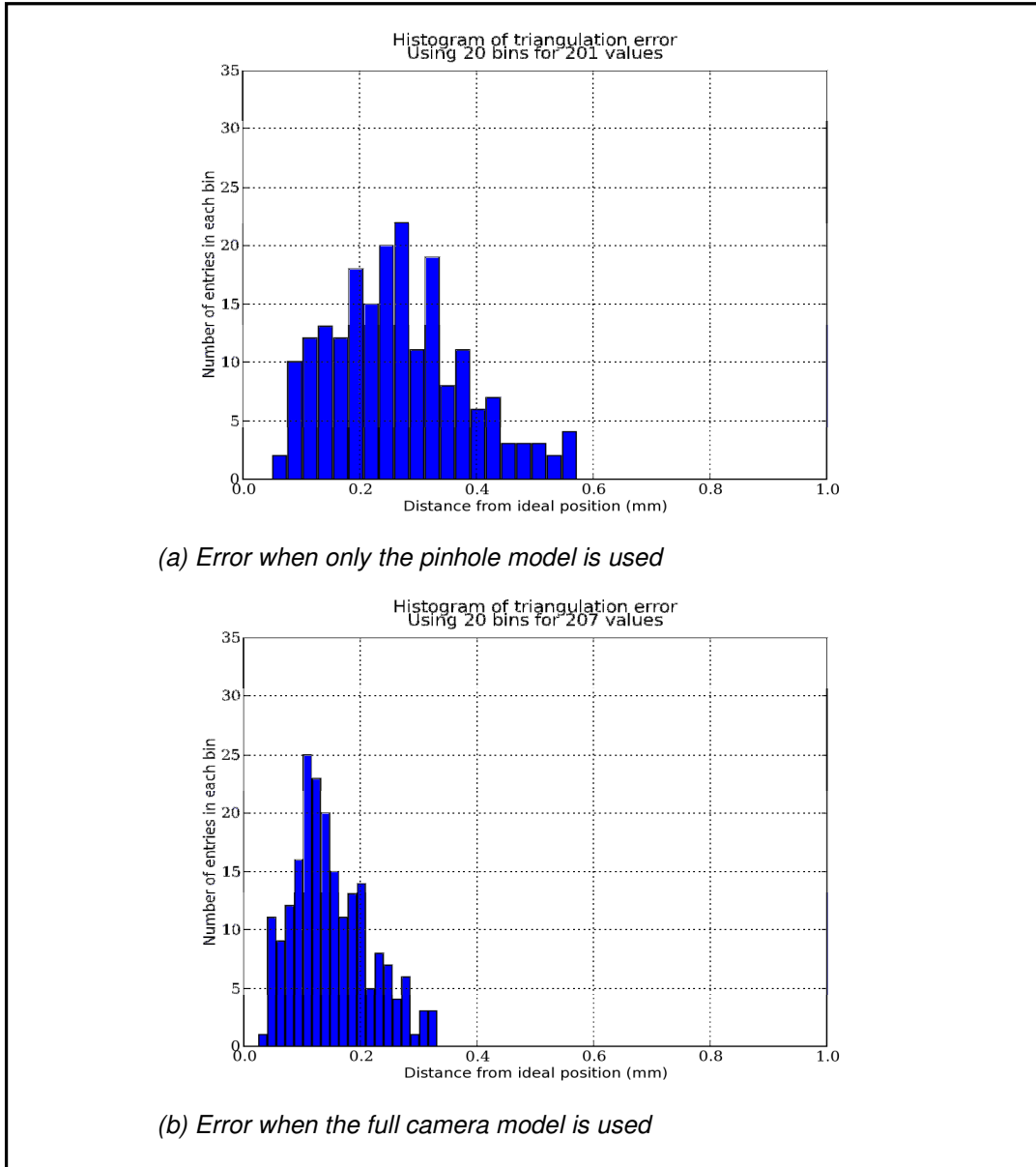


Figure 8-2: Triangulation errors for different camera models

8.1.3 Variability

For this test, all parameters are fixed to their standard values, but the calibration object is picked up and put down at approximately the same position. The fixed camera parameters are: a base-depth ratio of one and a full camera model. This test is repeated five times and results are shown in Table 8-5 and Table 8-6.

Test number	#1	#2	#3	#4	#5	Avg	Std. dev.
Colour camera							
Mean (pixels)	0.211	0.197	0.204	0.206	0.212	0.206	0.006
Std. deviation (pixels)	0.120	0.105	0.120	0.112	0.111	0.114	0.006
Monochrome camera							
Mean (pixels)	0.215	0.218	0.240	0.237	0.243	0.231	0.013
Std. deviation (pixels)	0.111	0.123	0.129	0.139	0.141	0.129	0.012

Table 8-5: Back-projection errors for variability study of calibrations

Test number	#1	#2	#3	#4	#5	Avg	Std. dev.
Mean (mm)	0.151	0.145	0.154	0.152	0.161	0.153	0.006
Std. deviation (mm)	0.066	0.071	0.071	0.078	0.078	0.073	0.005

Table 8-6: Triangulation errors for variability study of calibrations

All the standard deviation values are an order of magnitude less than the average values. The average values can therefore be said to fall within a range narrow enough to use them as good representative values in the other tests.

8.2 Correspondence Matching

The three methods developed for correspondence matching are now tested. The results for the laser tracking method, the square corner matching and the projected line crossings are all presented in Table 8-7.

The mean value is always very close to zero in each case because of the way the ideal plane has been fitted through the points. Not being very useful, the mean is therefore omitted from the results. The standard deviation remains very useful in determining the “flatness” of the plane and the consequent error in the system. Four times standard deviation (four-sigma) of the error is used as the final output value to evaluate these measurements.

Matching method	Square corners	Laser	Line crossings
Std. deviation (mm)	0.105	0.235	0.263
Four-sigma (mm)	0.419	0.940	1.052

Table 8-7: Comparison of matching method accuracy

The best results by far are given by the square corner method. This is understandable, because it extracts the matching coordinates much more accurately than the laser or line-crossing method. The laser-dot’s form is not very stable from frame to frame, making the calculation of its centre quite unpredictable. Lastly, the line-crossing method performs worst. Other methods than the weighted centroid calculation might have to be used to achieve greater accuracy with the line-crossing method.

Figure 8-3 compares the histograms of the error-sets for each of the methods using the same x-axis scale for comparison. It also shows the 3D visualisation of the coordinates. Note that for every method an area of about 210 x 240 mm is used. The spread of the histograms illustrate how the accuracy differs from method to method. The points in the 3D visualisations are fitted with a surface using a 2D Delaunay filter, one of the MayaVi visualisation package’s capabilities. Note that a lot less points are used in the line-crossing method than for the other two methods. This is because it is currently not very robust and is only capable of finding small amounts of correspondences correctly.

Note that for each case a different number of points are measured, which might have an affect on the final result. This does seem statistically unlikely though, because all the points are well distributed over the same surface area. It is also noted that the least number of points used is over a hundred, which is usually more than enough samples to sufficiently describe an error distribution.

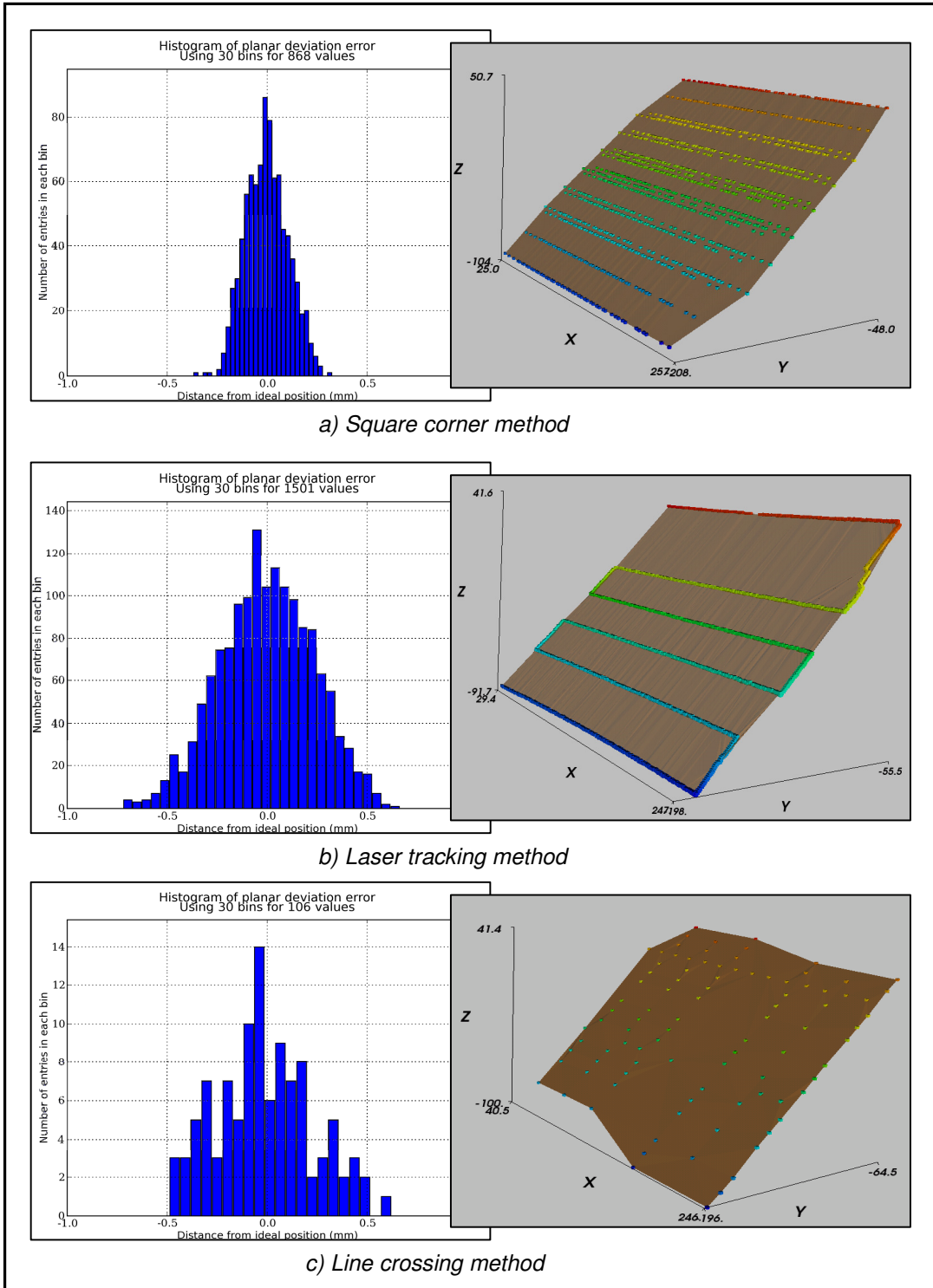


Figure 8-3: Error histograms and 3D visualisations for matching methods

8.3 A Practical 3D Measurement

The laser tracking method is used here to scan the profile of the bottle seen in Figure 8-4 (a), along with different presentations of the 3D data. Even though not the most accurate, this matching method is currently the only one capable of measuring more complex surfaces. This measurement is used for a qualitative evaluation only.

The point-cloud of the scanned profile consists of 15790 coordinates accumulated at about 12 fps. Points can of course only be constructed if the laser is visible in both images, which explains the loss of data around sharp bends. Note that the base-to-depth ratio used here is approximately 0.5 in order to increase the field of view common to both cameras.

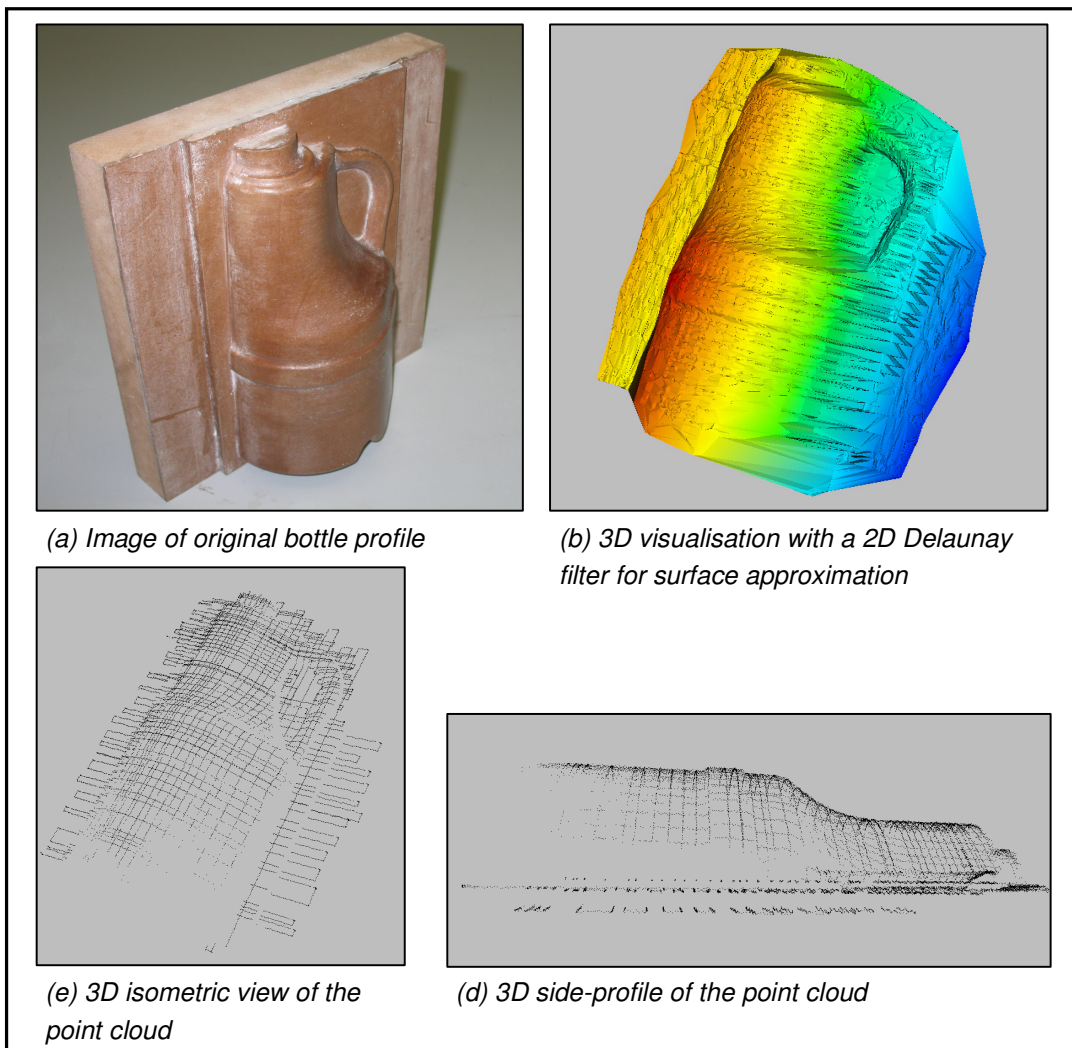


Figure 8-4: 3D Visualisation of scanned bottle profile

Chapter 9 Conclusions and Recommendations

9.1 Conclusions

A rapid optical measurement system has been developed and implemented for this project. It is capable of accumulating feature correspondences at 12 points per second with sub-millimetre accuracy. The accuracy achieved by calibration is better than 0.4 mm for a 235 x 190 x 95 mm volume, using only one image pair and an image resolution of 640 x 480 pixels.

An entire measurement, including calibration, correspondence matching and triangulation of a few thousand coordinates, can all be done in less than 20 minutes. The hardware components used in the system are relatively inexpensive, with a rough total cost of R 20 000 for the PC, cameras, projector, microcontroller and calibration object. These are the basic components needed for the implementation if only the projector is used for scanning. More importantly, all the software development for the system has been done using freely available software resources. The main advantage of this is that software developed using these resources can be commercialised without the need for expensive licence fees on the developer's end. A lot of the software in this project has been newly developed specifically for this measurement system, creating a flexible platform for future development. It has also allowed for a better understanding of the fundamental principles governing optical measurement techniques and can now act as an aid in further studies.

Most of the processes usually requiring time intensive user interaction in such a system has been automated using different image processing techniques in combination with the right hardware components. This includes the calibration phase as well as three different semi-automatic methods for solving the problem of rapid and accurate correspondence matching.

Finally, as a non-intrusive measurement technique capable of measuring complex smooth curvatures, it is uniquely suited for certain applications where even touch-probe devices may fail.

9.2 Shortcomings

The first issue to be addressed here is accuracy. The potential accuracy of the system as determined from the calibration process is below 0.4 mm. Achieving the same level of accuracy when rapidly accumulating coordinates is possible if the right coordinate extraction technique is used. This is illustrated by the square corner

detection with a four-sigma error of 0.412 mm when measuring a flat plane. The technique currently capable of practical measurements, however, achieves accuracies only slightly better than 1 mm. Even though there is an expected trade-off between the cost of such a system and its achievable accuracy, these results are not yet good enough for the system to be practically used for quality control.

The system is also limited to a very specific range of object sizes because of the calibration method that employs a 3D calibration object. The calibration object itself limits the achievable accuracy of the system and is not easy to manufacture.

Another limitation is the field of view. More accurate measurements are made for increased base-to-depth ratios, but this also causes features common to both images to be less.

9.3 Recommendations for Future Work

Even though the 3D calibration object has merit in a research environment, it is far from ideal when moving towards more practical implementation. For this reason it is recommended that either 2D calibration objects or self-calibration techniques should be employed. A calibration method is also needed for which the achievable accuracy is not dependent on the accuracy of the calibration object itself. This is exemplified by the bundle-adjustment techniques.

In order to solve the problem with the limited field of view, it will be necessary to implement some kind of data registration method. In this way an object can be scanned from different directions and the partial data-sets can be meshed together to form larger, more complete 3D coordinate sets. Another advantage is that overlapping data-sets can increase the overall measurement accuracy.

Finally, once complete data-sets for an object can be generated, it can be used for quality control by comparing it with the original computer designed model of that object.

References

- Armangué, X., Salvi, J. & Batlle, J. 2002. A Comparative Review of Camera Calibrating Methods with Accuracy Evaluation. *Pattern Recognition*, 1617-1635.
- Batista, J., Araújo, H. & de Almeida, A.T. 1998. *Iterative Multi-step Explicit Camera Calibration*. Proceedings: Sixth International Conference on Computer Vision, 15: 709-714.
- Boufama, B. & Habed, A. 2004. *Three-dimensional Structure Calculation: Achieving Accuracy Without Calibration*. *Image & Vision Computing*, 22.12: 1039-1049.
- Boyer, E. 2005. *Camera Calibration Using Silhouettes*. Research Report, Institut National de Recherche en Informatique et en Automatique, INRIA, 17 pages.
- Brown, D. 1972. *Calibration of Close-range Cameras*. ISP Congress, International Archives of Photogrammetry and Remote Sensing 19(5), unbound paper, 26 pages.
- Cao, X. & Foroosh, H. 2004. *Simple Calibration Without Metric Information Using an Isoceles Trapezoid*. , 2004. ICPR 2004. Proceedings, 17th International Conference on Pattern Recognition, ICPR, 1: 104-107.
- Cao, X. & Shah, M. 2005. *Camera Calibration and Light Source Estimation from Images with Shadows*. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR'05, 2: 918-923.
- Chi-Fang, L. & Chih-Yang, L. 1999. *A New Approach to High Precision 3-D Measuring System*. *Image and Vision Computing*, 17.11: 805-814.
- Clarke, T. A. & Fryer, J.G. (1998). *The Development of Camera Calibration Methods and Models*. *Photogrammetric Record* 16.91: 51-66.
- Csurka, G., Demirdjian, D., Ruf, A. & Horaud, R. 1998. *Closed-form Solutions for the Euclidean Calibration of a Stereo Rig*. Proceedings, 5th European Conference on Computer Vision, 426-444.
- Elter, M., Ernst, A. & Küblbeck, C. 2007. *A Passive 3D Scanner: Acquiring High-quality Textured 3D-models Using a Consumer Digital-camera*. Proceedings of the Second International Conference on Computer Vision Theory and Applications, VISAPP, 2: 311-316.

- Esteban, C. H. & Schmitt, F. 2003. *Using Silhouette Coherence for 3D Image-based Object Modeling Under Circular Motion*. Telekom Paris, Technical Report, 2003D011.
- Fitzgibbon, A. W., Cross, G. & Zisserman, A. 1998. *Automatic 3D Model Construction for Turn-table Sequences*. 3D Structure from Multiple Images of Large-Scale Environments, SMILE'98, 1506: 155-170.
- Foroosh, H., Balci, M. & Cao, X. 2005. *Self-calibrated Reconstruction of Partially Viewed Symmetric Objects*. Proceedings, International Conference on Acoustics, Speech, and Signal Processing, ICASSP, 2: 869-872.
- Fraser, C. S., Shortis, M. R. & Ganci, G. 1995. *Multi-sensor System Self-calibration*. Videometrics IV, SPIE 2598: 2-18.
- Fremont, V., Chelalli, R. 2002. *Direct camera calibration using two concentric circles from a single view*. Proceedings, International Conference on Artificial Reality and Telexistence, ICAT, 12:93–98.
- González, J. I., Gámez, J. C., Artal, C. G. & Cabrera, A. M. N. 2005. *Stability Study of Camera Calibration Methods*. VI Workshop en Agentes Físicos, (WAF)
- Gonzalez, R. C. & Woods R.E. 2002. *Digital Image Processing, International Edition*. Second Edition. New Jersey, Upper Saddle River: Prentice-Hall Inc.
- Gühring, J. 2000. *Dense 3-D Surface Acquisition by Structured Light Using Off-the-shelf Components*. Proceedings, SPIE 4309: 220–231.
- Guidi, G., Beraldin, J. & Atzeni, C. 2004. *High-Accuracy 3-D Modeling of Cultural Heritage: The Digitizing of Donatello's "Maddalena"*. IEEE Transactions on Image Processing, 13.3: 370-380.
- Guisser, L., Payrissat, R. & Castan, S. 2000. *PSGD: an Accurate 3D Vision System Using a Projected Grid for Surface Descriptions*. Image and Vision Computing, 18; 463-491.
- Hao, X. & Meyer, H. 2003. *Orientation and Auto-calibration of Image Triplets and Sequences*. ISPRS Conference on Photogrammetric Image Analysis, 34: 73-78

Hartley, R. and Zisserman, A. 2003. *Multiple View Geometry in Computer Vision*. Second Edition. United Kingdom, Cambridge: Cambridge University Press.

Heikkilä, J. & Silvén, O. 1997. *A Four-step Camera Calibration Procedure with Implicit Image Correction*. Proceedings, IEEE Computer Vision and Pattern Recognition, 1106-1112.

Heikkilä, J. 2000. *Geometric Camera Calibration Using Circular Control Points*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 22.10: 1066-1077. 22.10: 1066-1077.

Jiang, G., Quan, L. & Tsui, H. T. 2004. *Circular Motion Geometry Using Minimal Data*. IEEE Transactions on Pattern Analysis and Machine Intelligence. 26.6: 721-731.

Luong, Q. T. & Faugeras, O. D. 1997. *Self-calibration of a Moving Camera From Point Correspondences and Fundamental Matrices*. International Journal on Computer Vision, 22.3: 261-289.

Ma, Y., Soatto, S., Košecká, J. & Shankar Shastry, S. 2004. *An Invitation to 3-D Vision: From Images to Geometric Models*. New York: Springer-Verlag Inc.

Mallon, J., & Whelan, P. F. 2007. *Which Pattern? Biasing Aspects of Planar Calibration Patterns and Detection Methods*. Pattern Recognition Letters, 28.8: 921-930.

Mikhail, E. M., Bethel, J. S. & McGlone, J. C. 2001. *Introduction to Modern Photogrammetry*. USA: John Wiley & Sons, Inc.

Muller, N., De Kock E., Van Rooyen, R., Trauernicht, C. 2007. *A Stereophotogrammic System to Position Patients for Proton Therapy*. 2nd International Conference on Computer Vision Theory and Applications, VISAPP (2), 538-541.

Pappa, R. S., Giersch, L.R., and Quagliaroli, J. M., 2000, *Photogrammetry of a 5m Inflatable Space Antenna with Consumer Digital Cameras*, Technical Report: NASA-2000-tm210627.

Pedersini, F., Sarti, A., and Tubaro, S. 1999. *Accurate and Simple Geometric Calibration of Multi-camera Systems*. Signal Processing, 77.3: 309-334.

Peipe, J. & Tecklenburg, W. 2006. *Photogrammetric Camera Calibration Software – A Comparison*. ISPRS, In Proceedings, 5.6:266-272.

Remondino, F., & Fraser, C. 2006. *Digital Camera Calibration Methods: Considerations and Comparisons*. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, ISPRS, Vol. XXXVI: 266-272.

Scharstein, D. & Szelinski, R. 2003. *High-accuracy Stereo Depth Maps Using Structured Light*. IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 1: 195-202.

Schraml, S., Schön, P. & Milosevic, N. 2007. *Smartcam for Real-time Stereo Vision: Address-event Based Embedded Systems*. Proceedings of the Second International Conference on Computer Vision Theory and Applications, VISAPP, 2: 466-471.

Shortis, M. R., Clarke, T. A. & Short, T. 1994. *A Comparison of Some Techniques for the Subpixel Location of Discrete Target Images*. Videometrics II, Proc. SPIE 2350: 239–250.

Shortis, M. R., Snow, W. L. & Goad, W. K. 1995. *Comparative Geometric Tests of Industrial and Scientific CCD Cameras Using Plumb Line and Test Range Calibrations*. International Archives of Photogrammetry and Remote Sensing, 30(5W1): 53-59.

Stein, G. P. 1997. *Lens Distortion Calibration Using Point Correspondences*. IEEE Conference on Computer Vision and Pattern Recognition, 602–608.

Strum, P.F., Maybank, S.J., 1999. *On plane-based camera calibration: A general algorithm, singularities, applications*. IEEE Conference on Computer Vision and Pattern Recognition, 1: 432-437.

Triggs, B. 1998. *Autocalibration from Planar Scenes*. Proceedings, 5th European Conference on Computer Vision, ECCV, Vol. I: 89-105.

Trucco, E. & Verri, A. 1998. *Introductory Techniques for 3-D Computer Vision*. New Jersey, Upper Saddle River: Prentice-Hall, Inc.

Tsai, R. 1987. *A Versatile Camera Calibration Technique for High-accuracy 3-D Machine Vision Metrology Using Off-the-shelf TV Cameras and Lenses*. IEEE, Journal of Robotics and Automation, 3.4: 323-344.

- Van der Merwe, W. J. 2005. *Digital Photogrammetry: Close Range 3D Measurement of Objects Using A Consumer Digital Camera*. B.Eng. (Mechatronic) Final-year Project Report, Department of Mechanical & Mechatronic Engineering, University of Stellenbosch, South-Africa.
- Valkenburg, R. J. & McIvor, A.M. 1998. *Accurate 3D Measurement Using a Structured Light System*. *Image and Vision Computing*, 16: 99-110.
- Villa-Uriol, M. C., Chaudhary, B., Kuester, F., Hutchinson, T., & Bagherzadeh, N. 2004. *Extracting 3D from 2D: Selection Basis for Camera Calibration*. 7th IASTED International Conference on Computer Graphics and Imaging, 315–321.
- Zhang, G., Zhang, H. & Wong, K. K. 2006. *1D Camera Geometry and Its Application to Circular Motion Estimation*. *Proceedings, British Machine Vision Conference BMVC, Volume I*: 67-76.
- Zhang, S. & Huang, P. S. 2006. *High-resolution, Real-time Three-dimensional Shape measurement*. *Optical Engineering*, 45.12: 123601.
- Zhang, S. & Huang, P. S. 2006. *Novel Method for Structured Light System Calibration*. *Optical Engineering*, 45.8: 083601.
- Zhang, Z. 2000. *A flexible new technique for camera calibration*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22.11:1330-1334.

Appendix A Pseudo Code

This section gives the pseudo code for some of the most essential algorithms developed for the thesis.

A.1. Sub-pixel Line Detection

FUNCTION NAME:

findSubPixProjectorLineCrossingPts

DESCRIPTION:

Finds the sub-pixel coordinates of the crossing points formed by two images of the same stationary object, the one with vertically projected and the other with horizontally projected light stripes.

INPUTS:

Image1, Image2: images containing the vertically and horizontally projected lines respectively.

GammaCorrectionFlag: does gamma-correction on the images if set.

ThresholdFraction: a value between 0 and 1 that sets the threshold as a fraction of the maximum intensity of the image. With a higher value, more points can be found, but also more noisy points.

MaximumNumberOfPoints: the maximum number of points that must be found.

OUTPUTS:

Coordinates: the coordinates of the crossing points.

PSEUDO CODE:

*IF Image1 or Image2 is in the range [0,255]:
SCALE images to range [0, 1]*

*IF GammaCorrectionFlag is = 1:
APPLY GAMMA CORRECTION to Image1 and Image2*

CALCULATE ImageDy; the derivative of Image1 using a 3x3 Sobel operator in the y-direction

CALCULATE ImageDx; the derivative of Image2 using a 3x3 Sobel operator in the x-direction

CALCULATE ImageSum; the per element summation of Image1 and Image2

SCALE ImageSum to range [0, 1]

```

FIND IntensityMin and IntensityMax; the minimum and
maximum intensity values of ImageSum respectively

INITIALISE CurrentMax = 1.0; the maximum intensity value
found in ImageSum in one run of the while-loop
INITIALISE PointCounter = 0; the counter for checking
that the maximum number of points have not yet been
found
INITIALISE CoordinateList = empty list; a list to
contain the sub-pixel coordinates of the crossing
points

WHILE CurrentMax > (ThresholdFraction*IntensityMax) and
PointCounter < MaximumNumberOfPoints:

    INCREASE PointCounter by 1

    FIND CurrentMin and CurrentMax; the current minimum
and maximum intensity values of ImageSum
respectively

    CALCULATE Square; the square position, width and
height as a function of the size of the Sobel
operator and the location of CurrentMax, to be
used to extract a region of interest in ImageSum
containing the current crossing point

    EXTRACT ImageSquare; the region of interest in
ImageSum using Square

    CALCULATE CenterX and CenterY; the center-of-mass
coordinates of ImageSquare with respect to the
origin of the original image, ImageSum

    STORE CenterX and CenterY in CoordinateList

    ELIMINATE the intensity values contained in the
current region of interest in ImageSum by
setting them to 0.

```

A.2. Correspondence Matching Using Corner Detection

FUNCTION NAME:

getSqrCornerMatches

DESCRIPTION:

Finds and accumulates sub-pixel coordinates of the corners of three squares projected onto an object (currently tested only a flat surface). The corners are found for a stereo pair of images and then matched.

INPUTS:

FileDirectory: the directory in which to save the matching coordinates.

MatchFile: name of the file to save the matches in.

Trigger: flag to set the external triggering of the cameras on or off with a 1 or 0 respectively.

OUTPUTS:

CornerMatches: the matching coordinates.

PSEUDO CODE:

INITIALISE the digital video cameras

INITIALISE ImageList = [Image1,Image2]; a list with two image structures for temporarily storing images grabbed by the cameras

CREATE Window1, Window2; two windows for displaying the images from the cameras

CREATE Image; an image structure in which to temporarily copy a camera image for corner detection

INITIALISE MaxCount, Quality, MinDistance; the maximum number of corners that can be found, the quality the corners must have and the minimum allowed distance between corners respectively

INITIALISE MatchList = [[],[]]; a list to contain the matching coordinates of the square corners in each image

INITIALISE KeyPressed = -1; the variable to store the current value of a keyboard input

```

WHILE KeyPressed <> 'x':

    GET KeyPressed; read the current key input from the
    keyboard

    GRAB Image1 and Image2; the current camera images
    stored in ImageList
    GET TimeStamp; a list with the current timestamp of
    each camera

    INITIALISE Synced = 1; the flag indicating if the
    cameras are synchronised or not

    IF TimeStamp indicates the cameras are synchronised
    and Trigger = 1:
        SET Synced = 1
    ELSE:
        SET Synced = 0
    ELSE IF Trigger = 0:
        SET Synced = 1; force the synchronisation flag
        if the cameras are not triggered by an
        external trigger for synchronisation

    INITIALISE FoundAll0 = 0, FoundAll1 = 0; flags
    indicating that all corners have been found in
    the first and second camera respectively

    ##Repeat for both cameras
    FOR i in [0,1]:

        IF KeyPressed = 's' and Synced = 1:
            IF ImageList[i] is a colour image:
                CONVERT ImageList[i] to grayscale Image
            ELSE:
                COPY ImageList[i] to Image

        FIND Corners; a list of discrete pixel
        coordinates of the strong corner
        features in Image (use Open CV function
        cvFindGoodFeaturesToTrack)

        IF 12 corners were found in Corners:

            CALCULATE CornersNew; the sub-pixel
            coordinates using Corners (use Open
            CV function cvFindCornersSubPix)

            CONVERT CornersNew to TempArr; an array
            used to sort the corners

            SORT TempArr; coordinates sorted in two

```

*columns from left to right, each
column sorted top to bottom*

```
IF i = 0:  
    APPEND MatchList[0] with TempArr;  
    SET AllFound0 flag to 1  
IF i = 1:  
    APPEND MatchList[1] with TempArr;  
    SET AllFound1 flag to 1  
  
IF AllFound0 =1 and AllFound1 =1:  
    PRINT 'Saved'; indicate on-screen  
    that all corners were found in  
    both images
```

*DRAW Corners in ImageList[i]; the
current image*

```
IF i = 0: DISPLAY imageList[0] in Window1  
IF i = 1: DISPLAY imageList[1] in Window2
```

```
IF AllFound0 = 0 and AllFound1 =1:  
    REMOVE current corner set from MatchList[1]  
IF AllFound0 = 1 and AllFound1 = 0:  
    REMOVE current corner set from MatchList[0]
```

POMPT UserInput; *prompt user for input choosing to save
the matchlist or not*

```
IF UserInput = 'y':  
    SAVE MatchList in the path specified by  
    FileDirectory and MatchFile
```


A.3. Correspondence Matching by Tracking a Moving Laser Dot

FUNCTION NAME:

getLaserMatches

DESCRIPTION:

Finds and accumulates sub-pixel coordinates of the corners of three squares projected onto an object (currently tested only a flat surface). The corners are found for a stereo pair of images and then matched.

INPUTS:

FileDirectory: the directory in which to save the matching coordinates.

MatchFile: name of the file to save the matches in.

OUTPUTS:

LaserMatches: the matching coordinates.

PSEUDO CODE:

INITIALISE the digital video cameras

INITIALISE ImageList = [Image1,Image2]; a list with two image structures for temporarily storing images grabbed by the cameras

CREATE Window1, Window2; two windows for displaying the images from the cameras

INITIALISE ImgGray, ImgDiff, ImgTemp; three image lists each containing two empty image structures, for grayscale image conversions, differenced images and temporary images respectively

INITIALISE LaserLoc = [p1,p2]; a list with two empty coordinate structures to contain the centre coordinates of the currently detected laser-dot in each image

INITIALISE MatchList = [[],[]]; an empty list to contain the image correspondences of the laser-dot

INITIALISE Found = [0,0]; a list of flags to check if laser has been found in both images

INITIALISE Accumulate = 0; flag for toggling

accumulation of laser-dot coordinates

```
INITIALISE KeyPressed = -1; to contain the current
value of the keyboard input

WHILE KeyPressed <> 'x':

    GET KeyPressed; read the current key input from the
    keyboard if any

    IF KeyPressed = 'a':
        PRINT 'Toggling Accumulation'; indicate
        on-screen that accumulation has been toggled

        TOGGLE the Accumulation flag on or off

    IF KeyPressed = 's' and MatchList is not empty:
        SAVE the MatchList correspondences to the path
        indicated by FileDirectory and MatchFile

    ##DO FOR BOTH CAMERAS:
    FOR i in [0,1]:
        IF ImageList[i] is a colour image:
            CONVERT ImageList[i] to ImgGray[i] grayscale
            image
        ELSE:
            COPY ImageList[i] to ImageGray[i]

            COPY ImageList[i] to ImgTemp[i]; the previous
            iteration's camera images temporarily stored
            in order to calculate the difference

    GRAB current cameras images and store in ImageList
    GET TimeStamp; list with current timestamp of each
    camera

    IF TimeStamp indicates the images are synchronised:
        PRINT 's'; on-screen confirmation of
        synchronisation

        RESET Found = [0,0]; flags must be reset every
        time a new image is grabbed
        ##DO FOR EACH CAMERAS:
        FOR i in [0,1]:
            IF ImageList[i] is a colour image:
                CONVERT ImageList[i] to ImageGray[i]
                grayscale image
            ELSE:
                COPY ImageList[i] to ImageGray[i]

            CALCULATE the difference in ImgGray[i] and
```

```

    ImgTemp[i] and store it in ImgDiff[i]

    FIND ImgMax and LaserLoc[i]; the maximum
    value and location of the maximum value
    in ImgDiff[i] respectively

    IF ImgMax is > some threshold:
        Found[i] = 1

    IF Found[0] = 1 and Found[1] = 1 and
        Accumulate = 1:

        INITIALISE LaserCoords = [None, None]; a list
        to contain the temporary coordinates for
        the laser-dot in the 1st and 2nd camera
        image

        ##DO FOR BOTH CAMERA IMAGES:
        FOR i in [0,1]:
            EXTRACT ImgSub; a small ROI around the
            current location given by
            LaserLoc[i]

            CALCULATE ImgBW; the thresholded binary
            image of ImgSub

            IF ImgSub has only 1 connected element:
                CALCULATE LaserCoords[i]; the centre
                of the connected element in
                ImgSub using a 2D centre of mass
                calculation (use SciPy's ndimage
                module's center_of_mass
                function)

            IF both LaserCoords entries exist:
                APPEND MatchList with the LaserCoords
                entries

    DISPLAY ImageList images in Window1 and Window2

    RETURN MatchList

```

A.4. Automatic Detection of Calibration Grid Corners

The pseudo code for the automatic sub-pixel detection of all corners on the calibration grid consists of *###* sequentially implemented functions. The first is to detect all square objects and their centers and fit four-sided polygons to the objects where possible. The second is to sort all the centers for correspondence matching in two images. The third step is to link the detected centers with the available squares (polygons) and the fourth to approximate polygons for the square objects for which only centers could be calculated. The fifth and final step is to approximate the sub-pixel coordinates for the corners of each square.

The pseudo code for steps two through four will not be given. They are explained conceptually in section *###*. Steps one and five are considered the most important in the auto-detection process and are given here.

Step 1: Detect All Square Objects

FUNCTION NAME:

find3DgridSqrs

DESCRIPTION:

Finds a grid of square objects on a 3D calibration grid consisting of well contrasted square surfaces on two perpendicular surfaces.

INPUTS:

Image: the image of the calibration grid with all squares visible and well contrasted with the background.

OUTPUTS:

Squares: a list of corner-coordinates in sets of four, each set defining the four corners of a approximated square object. Not all square objects will definately have such a polygon fitted to it.

SquareCenters: a list of coordinates of the approximated center of each square object. Each square object will have a center coordinate, regardless of whether a four sided polygon was fitted to it succesfully or not.

PSEUDO CODE:

*IF Image is a colour image:
 CONVERT Image colour to Image grayscale*

FILTER Image for noise suppression; use Open CV function

cvPyrDown and cvPyrUp

APPLY Canny edge detection to Image; use Open CV
function cvCanny

APPLY closing morphological filter on Image using a 5x5
mask; *closes gaps in the edge-detected image*

APPLY dilating morphological filter on Image using a 3x3
mask in a four-connected arrangement; *improves the
form of the square objects for square fitting*

FIND Contours in Image; *all the contours of separate
binary objects (use Open CV function cvFindContours)*

COPY Image to ImageTemp; *a temporary image of the
contours*

FILL all enclosed contour objects in Contours and draw
in Image

SUBTRACT ImageTemp from Image; *this leaves only the
objects that had enclosed edges, removing spurious
artifacts like lines*

FIND Contours in Image; *repeat contour finding after
first iteration of unwanted object removal*

REMOVE all contour elements in Image that is shorter
than some fraction of the maximum contour in
Contours; *assuming the longest contour is that of
one of the squares, this step removes objects too
small to qualify as squares*

FIND Contours; *all the new contours in Image after
smaller objects have been rejected*

FILL any remaining holes in Image

FIND Contours; *final contour retrieval from Image*

INITIALISE SqrCounter; *counter for the number of square
centers calculated*

INITIALISE ContourLengths; *list to contain the length
of each contour*

INITIALISE SquareCenters; *list to contain the
coordinates of each calculated square center*

###DO FOR ALL CONTOURS:
FOR Contour in Contours:

```

APPROXIMATE ApproxContour; a polygon fitted to
    Contour, the contour of the current iteration
    (use Open CV function cvApproxPoly)

IF (ApproxContour has 5 or 6 sides) and
    ApproxContour is convex:

    INCREMENT SqrCounter by 1
    CALCULATE CenterOfMass; the coordinates of the
        center of mass for ApproxContour
    APPEND SquareCenters list with CenterOfMass

IF ApproxContour has 4 sides AND
    ApproxContour is convex AND
    the maximum angle between two sides is below
    a certain threshold:

    APPEND Squares with ApproxPoly; store the
        polygon of the current iteration in Squares
        as a detected square

    CALCULATE SquareCenter; the center coordinate
        of ApproxPoly as the intersection of the
        lines formed by opposite corners of the
        polygon

    APPEND SquareCenters with SquareCenter

RETURN Squares, SquareCenters

```

Step 5: Deriving Sub-pixel Coordinates for the Grid Squares

FUNCTION NAME:

findAllSubPixSqrCorners2

DESCRIPTION:

Derives the sub-pixel coordinates of a 3D calibrations grid's square corners. The order and approximated position of each square object in the image must pre-determined. The function fits lines to each of the four sides of a square object and calculates the coordinates of the line intersections as the derived corner coordinates.

INPUTS:

Image: the grayscale image of the 3D calibration grid. All squares must be visible and well-contrasted. This is the same image used in the find3DgridSqrS

function.

Squares: a list of coordinate sets, each set containing four coordinates describing the approximated corners of a four sided polygon fitted to each square object. Same size as SquareCenters after adding missing squares.

SquareCenters: coordinate list of the sorted centres of each square.

LinkIndex: an array of index numbers linking Squares with SquareCenters. Being separate lists, only SquareCenters was sorted correctly and needs a index number to connect it to the correct entry in the unsorted SquareCenters list.

KernelSize: the size of the convolution kernel used to create the derivative images for edge detection.

OUTPUTS:

Xcoordinates, Ycoordinates: the sub-pixel x and y image coordinates of the derived square corners. Each four consecutive coordinates define a square.

PSEUDO CODE:

CREATE Kernel; the 1D convolution kernel used to calculate the image derivatives. Use the create1DderivativeKernel function and the KernelSize input.

INITIALISE Xcorners; list to contain the x coordinates of the derived corners.

INITIALISE Ycorners; list to contain the y coordinates of the derived corners.

###DO FOR ALL SQUARES:

FOR i in range(all available squares):

EXTRACT ImageSub; the ROI around the current square in Image at coordinate SquareCenters[i]

FIND VerticalEdgePts with getSubPixLinePts function

FIND HorizontalEdgePts with getSubPixLinePts function

CALCULATE Xcorner,Ycorner; two lists, each with the four x and y coordinates of the intersections of the lines formed by the points in VerticalEdgePts and HorizontalEdgePts. This will then be the derived corner coordinates for the current square. Use getSubPixSqrCorners function.

APPEND Xcorners and Ycorners with Xcorner and Ycorner respectively

RETURN Xcorners, Ycorners

5.1 Finding vertical and horizontal edge points

FUNCTION NAME:

getSubPixLinePts

DESCRIPTION:

Calculates the sub-pixel position of of strong vertical or horizontal edges in an image. For a horizontal search direction, vertical edges are detected for each row in the image. For a vertical search direction, horizontal edges are detected for each column in the image.

INPUTS:

Image: a grayscale image with strong vertical or horizontal edge features

Kernel: a 1D derivative convolution kernel created using the create1DderivativeKernel function

BorderCut: fraction of the image to ignore around its border

ThresholdFraction: a multiplication fraction multiplied with the maximum of the intensity peaks found in a row or column of the derivative image. The intensity peaks indicate edge features and the fraction is used to determine the threshold below which edges will be rejected.

Direction: either a 0 to indicate a horizontal search direction for vertical lines or 1 for a vertical search of horizontal edges.

OUTPUT:

LinePositionsA, LinePositionsB: lists containing either the column or row indexes (depending on the search direction) of only the columns or rows in which edge points were detected.

EdgeCoordinatesA, EdgeCoordinatesB: a list of lists, where each list contains all the line points found for every row or column in the image, if any.

NOTE: the A and B in each of these variables indicate edges detected going from dark to light (A) and from light to dark (B). This can be used later on to simplify segmentation and the separation of different lines.

PSEUDO CODE:

GET KernelSize; *the number of elements in Kernel*


```

IF Direction = 1:
    CHANGE Image to the transpose of Image; flips the
        image by 90 degrees
ELSE IF Direction = 0:
    Image remains unchanged

CALCULATE the derivative image ImageDx by convolving
    Image with Kernel; the derivative image should now
        contain positive and/or negative peaks depending on
        whether edges went from low to high intensity and/or
        vice versa

GET Height, Width; the height and width of ImageDx

CALCULATE MaxThreshold as: ThresholdFraction*(maximum
    intensity value in ImageDx); the threshold value
        below which positive intensity values are rejected
        as edges
CALCULATE MinThreshold as: ThresholdFraction*(minimum
    intensity value in ImageDx); the threshold value
        above which negative intensity values are rejected
        as edges

INITIALISE LinePositionsA and LinePositionsB as
    empty lists
INITIALISE EdgeCoordinatesA and EdgeCoordinatesB
    as empty lists

INITIALISE StepX; a list of sequential index values for
        every column searched in a row

INITIALISE RangeY using BorderCut and Height; list of
        sequential index values for the rows to be searched

###DO FOR EVERY ROW EXTRACTED:
FOR i in RangeY:

    EXTRACT Line; the i'th row from ImageDx

    ###FIND SUB-PIXEL CURVE PEAKS FOR EDGES GOING FROM
    ###DARK TO LIGHT:
    INITIALISE SubPixelList as empty list; a list to
        contain all sub-pixel positions of edges found
        in Line

    INITIALISE EdgesLeft as 1; a flag to indicate
        whether there are peaks left in Line that
        qualify as edges

    WHILE EdgesLeft = 1:

```

```

GET MaxValue; the maximum intensity value in
Line

###FIND PEAKS ABOVE THE THRESHOLD:
IF MaxValue >= MaxThreshold:
    GET MaxPosition; the position of MaxValue
in Line
    GET StartIndex, StopIndex; the indexes
between which the discrete curve points
containing MaxVal must be extracted.
Use KernelSize to determine how many
points must be extracted on each side
of MaxPosition

    EXTRACT CurveDiscrete using StartIndex and
    StopIndex; the discrete curve that
also contains MaxVal, formed by the
intensity values in Line

    GET ValuesX from StepX using StartIndex and
    StopIndex; the column positions for the
CurveDiscrete entries

    CALCULATE Coeffs; the coefficients of a
parabola fitted to CurveDiscrete and
ValuesX using a least-squares estimation

    CALCULATE MaxMidPoint using Coeffs; the
exact position of the maximum value, or
turning point, of the fitted parabola

    APPEND SubPixelList with MaxMidPoint

    REMOVE CurveDiscrete values from Line;
    this is done so that the next peak can
be found in the following iteration

ELSE:
    EdgesLeft = 0
    If SubPixelList is not empty:
        APPEND LinePositionsA with i; the
current row index
        APPEND EdgeCoordinatesA with
        SubPixelList

###FIND SUB-PIXEL CURVE PEAKS FOR EDGES GOING FROM
###LIGHT TO DARK:
INITIALISE SubPixelList as empty list; a list to
contain all sub-pixel positions of edges found
in Line

```

```

INITIALISE EdgesLeft as 1; a flag to indicate
           whether there are peaks left in Line that
           qualify as edges

WHILE EdgesLeft = 1:

    GET MinValue; the minimum intensity value in
       Line

    ###FIND PEAKS BELOW THE THRESHOLD:
    IF MinValue <= MinThreshold:
        GET MinPosition; the position of MinValue
           in Line
        GET StartIndex, StopIndex; the indexes
           between which the discrete curve points
           containing MinVal must be extracted.
           Use KernelSize to determine how many
           points must be extracted on each side
           of MinPosition

        EXTRACT CurveDiscrete using StartIndex and
           StopIndex; the discrete curve that
           also contains MinVal, formed by the
           intensity values in Line

        GET ValuesX from StepX using StartIndex and
           StopIndex; the column positions for the
           CurveDiscrete entries

        CALCULATE Coeffs; the coefficients of a
           parabola fitted to CurveDiscrete and
           ValuesX using a least-squares estimation

        CALCULATE MinMidPoint using Coeffs; the
           exact position of the minimum value, or
           turning point, of the fitted parabola

        APPEND SubPixelList with MinMidPoint

        REMOVE CurveDiscrete values from Line;
           this is done so that the next peak can
           be found in the following iteration

    ELSE:
        EdgesLeft = 0
        If SubPixelList is not empty:
            APPEND LinePositionsB with i; the
               current row index
            APPEND EdgeCoordinatesB with
               SubPixelList

```

RETURN EdgeCoordinatesA, EdgeCoordinatesB,
LinePositionsA, LinePositionsB

5.2 Fitting lines to edge points and calculating intersections

FUNCTION NAME:

getSubPixSqrCorners

DESCRIPTION:

Use vertical and horizontal edge coordinates of the edges of a square object from a calibration grid to fit lines to the edges. Return the intersections of these lines that will then be the derived corner positions for each square object.

INPUTS:

VerticalEdgeInfo, HorizontalEdgeInfo: Two lists, each containing the four variables returned from the getSubPixLinePts function. It is assumed that each list contains only the edge coordinates of a square object from the calibration grid.

OUTPUTS:

Xcorners, Ycorners: the four x and y coordinates of the square corners, sorted clockwise from the top left corner.

PSEUDO CODE:

REMOVE unwanted line coordinates from VerticalEdgeInfo
REMOVE unwanted line coordinates from HorizontalEdgeInfo

INITIALISE LinesX as empty list; to contain the x
coordinates of all four edges of the square
EXTRACT LinesX from VerticalEdgeInfo and
HorizontalEdgeInfo

INITIALISE LinesY as empty list; to contain the y
coordinates of all four edges of the square
EXTRACT LinesY from VerticalEdgeInfo and
HorizontalEdgeInfo

INITIALISE CoeffList as empty list; to contain the
Coefficients for each line

###DO FOR ALL FOUR EDGES OF A SQUARE:

FOR i in range(4):
CALCULATE Coeffs for the i'th entry in LinesX and
LinesY; a list containing the three coefficients
of the line equation $0 = ax + by + c$, where

```
        a, b and c are the coefficients. Use the  
        fitPolynomial function.  
APPEND CoeffList with Coeffs  
  
INITIALISE Xcorners and Ycorners as empty lists  
  
CALCULATE Xcorners, Ycorners using Coeffs; the x and  
        y coordinates of the intersections of the lines  
        described by the coefficients in CoeffList.  
        Intersections calculated in the order: top left, top  
        right, bottom right and bottom left.  
  
return Xcorners, Ycorners
```

Appendix B Test Results

B.1. Base-to-depth Ratio

		Base-to-depth ratio: 1										
				Experim. run:								
				1	2	3	4	5	AVG	STD		
COLOUR CAMERA	Mean	0.203	0.221	0.229	0.219	0.223	0.219	0.010	Back-projection error (pixels)			
	Std	0.101	0.119	0.116	0.115	0.129	0.116	0.010				
	RMS	0.227	0.251	0.257	0.247	0.257	0.248	0.012				
	Max	0.486	0.490	0.500	0.555	0.639	0.534	0.065				
MONO-CHROME CAMERA	Mean	0.213	0.217	0.226	0.232	0.238	0.225	0.010				
	Std	0.105	0.120	0.135	0.124	0.130	0.123	0.011				
	RMS	0.237	0.248	0.263	0.263	0.271	0.256	0.014				
	Max	0.508	0.562	0.592	0.635	0.556	0.571	0.047				
		Mean	0.148	0.155	0.166	0.161	0.157	0.157	0.007	Triangulation error (mm)		
		Std	0.068	0.076	0.082	0.081	0.077	0.077	0.005			
		RMS	0.163	0.173	0.186	0.180	0.175	0.175	0.009			
		Max	0.351	0.371	0.378	0.359	0.348	0.361	0.013			
		Mean+3*sigma	0.353	0.383	0.412	0.404	0.388	0.388	0.023			

		Base-to-depth ratio: 0.5										
				Experim. run:								
				1	2	3	4	5	AVG	STD		
COLOUR CAMERA	Mean	0.208	0.212	0.240	0.202	0.209	0.214	0.015	Back-projection error (pixels)			
	Std	0.107	0.110	0.143	0.096	0.115	0.114	0.018				
	RMS	0.234	0.239	0.279	0.224	0.239	0.243	0.021				
	Max	0.502	0.481	0.770	0.441	0.511	0.541	0.131				
MONO-CHROME CAMERA	Mean	0.279	0.276	0.265	0.265	0.276	0.272	0.007				
	Std	0.145	0.137	0.122	0.150	0.136	0.138	0.011				
	RMS	0.314	0.308	0.292	0.304	0.308	0.305	0.008				
	Max	0.772	0.576	0.578	0.767	0.653	0.669	0.097				
		Mean	0.229	0.198	0.217	0.216	0.211	0.214	0.011	Triangulation error (mm)		
		Std	0.137	0.097	0.122	0.124	0.112	0.118	0.015			
		RMS	0.267	0.221	0.249	0.249	0.239	0.245	0.017			
		Max	0.709	0.397	0.549	0.548	0.510	0.543	0.112			
		Mean+3*sigma	0.640	0.489	0.583	0.588	0.547	0.569	0.056			

B.2. Camera Model Complexity

		Camera model: DLT					AVG	STD	
COLOUR CAMERA	Mean	0.338	0.349	0.346	0.386	0.348	0.353	0.019	Back-projection error (pixels)
	Std	0.191	0.191	0.178	0.206	0.191	0.191	0.010	
	Max	0.945	0.985	0.969	1.164	0.981	1.009	0.088	
MONO-CHROME CAMERA	Mean	0.432	0.411	0.427	0.449	0.435	0.431	0.014	Triangulation error (mm)
	Std	0.227	0.223	0.231	0.258	0.245	0.237	0.014	
	Max	1.091	1.241	1.266	1.394	1.087	1.216	0.129	
	Mean	0.226	0.271	0.269	0.283	0.279	0.266	0.023	Triangulation error (mm)
	Std	0.114	0.132	0.114	0.127	0.124	0.122	0.008	
	Max	0.574	0.698	0.610	0.725	0.587	0.639	0.068	
	Mean+3*sigma	0.568	0.667	0.611	0.664	0.651	0.632	0.042	

		Camera model: k1					AVG	STD	
COLOUR CAMERA	Mean	0.228	0.215	0.217	0.230	0.227	0.223	0.007	Back-projection error (pixels)
	Std	0.129	0.117	0.129	0.117	0.118	0.122	0.006	
	Max	0.659	0.501	0.566	0.539	0.542	0.561	0.059	
MONO-CHROME CAMERA	Mean	0.236	0.239	0.252	0.257	0.256	0.248	0.010	Triangulation error (mm)
	Std	0.134	0.142	0.138	0.144	0.153	0.142	0.007	
	Max	0.594	0.704	0.603	0.652	0.650	0.641	0.044	
	Mean	0.160	0.154	0.170	0.162	0.168	0.163	0.006	Triangulation error (mm)
	Std	0.075	0.077	0.085	0.077	0.081	0.079	0.004	
	Max	0.346	0.357	0.395	0.341	0.346	0.357	0.022	
	Mean+3*sigma	0.385	0.385	0.425	0.393	0.411	0.400	0.018	

		Camera model:					k1k2		AVG	STD	
COLOUR CAMERA	Mean	0.220	0.208	0.215	0.218	0.220	0.216	0.005	Back-projection error (pixels)		
	Std	0.122	0.111	0.125	0.112	0.111	0.116	0.007			
	Max	0.601	0.478	0.537	0.515	0.506	0.527	0.046			
MONO-CHROME CAMERA	Mean	0.224	0.223	0.239	0.244	0.247	0.235	0.011	Triangulation error (mm)		
	Std	0.119	0.132	0.130	0.139	0.144	0.133	0.010			
	Max		0.688	0.572	0.616	0.612					
	Mean	0.152	0.148	0.159	0.156	0.163	0.156	0.006	Triangulation error (mm)		
	Std	0.065	0.074	0.072	0.076	0.079	0.073	0.005			
	Max	0.330	0.362	0.345	0.348	0.346	0.346	0.011			
	Mean+3*sigma	0.347	0.370	0.375	0.384	0.400	0.375	0.019			

		Camera model:					k1k2c		AVG	STD	
COLOUR CAMERA	Mean	0.211	0.197	0.204	0.206	0.212	0.206	0.006	Back-projection error (pixels)		
	Std	0.120	0.105	0.120	0.112	0.111	0.114	0.006			
	Max	0.552	0.461	0.528	0.511	0.510	0.512	0.033			
MONO-CHROME CAMERA	Mean	0.215	0.218	0.240	0.237	0.243	0.231	0.013	Triangulation error (mm)		
	Std	0.111	0.123	0.129	0.139	0.141	0.129	0.012			
	Max	0.523	0.644	0.562	0.606	0.603	0.588	0.046			
	Mean	0.151	0.145	0.154	0.152	0.161	0.153	0.006	Triangulation error (mm)		
	Std	0.066	0.071	0.071	0.078	0.078	0.073	0.005			
	Max	0.333	0.342	0.325	0.340	0.337	0.335	0.007			
	Mean+3*sigma	0.349	0.358	0.367	0.386	0.395	0.371	0.019			

B.3. Planar Deviation

The camera calibration triangulation error (mm):

Mean	0.161
Std	0.074
Max	0.338
Mean+3*std	0.383

Planar deviation error (mm):

	Square corners	Line Crossing	Laser
Nr. of points:	868	106	1501
Std	0.105	0.263	0.235
Min	-0.367	-0.486	-0.718
Max	0.325	1.100	0.666
Min/max range	0.692	1.586	1.384
4*std	0.419	1.052	0.940