# Design and Application of an Automated System for Camera Photogrammetric Calibration

**Jason de Villiers**
MEng (Electronic) Univ. of Pretoria

A thesis submitted to the Department of Electrical Engineering, University of Cape Town, in fulfilment of the requirements for the degree of

**Doctor of Philosophy**

at the

**University of Cape Town**

December 2014

# Declaration

I declare that this thesis is my own work. It is being submitted to the Department of Electrical Engineering, University of Cape Town, in fulfilment of the requirements for the degree of Doctor of Philosophy. It has not been submitted before for any degree or examination in any other university.

Signature of Author...........................................................

Cape Town
31 December 2014

# Abstract

This work presents the development of a novel Automatic Photogrammetric Camera Calibration System (APCCS) that is capable of calibrating cameras, regardless of their Field of View (FOV), resolution and sensitivity spectrum. Such calibrated cameras can, despite lens distortion, accurately determine vectors in a desired reference frame for any image coordinate, and map points in the reference frame to their corresponding image coordinates. The proposed system is based on a robotic arm which presents an interchangeable light source to the camera in a sequence of known discrete poses. A computer captures the camera's image for each robot pose and locates the light source centre in the image for each point in the sequence.

Careful selection of the robot poses allows cost functions dependant on the captured poses and light source centres to be formulated for each of the desired calibration parameters. These parameters are the Brown model parameters to convert from the distorted to the undistorted image (and vice versa), the focal length, and the camera's pose. The pose is split into the camera pose relative to its mount and the mount's pose relative to the reference frame to aid subsequent camera replacement. The parameters that minimise each cost function are determined via a combination of coarse global and fine local optimisation techniques: genetic algorithms and the Leapfrog algorithm, respectively.

The real world applicability of the APCCS is assessed by photogrammetrically stitching cameras of differing resolutions, FOVs and spectra into a single multispectral panorama. The quality of these panoramas are deemed acceptable after both subjective and quantitative analyses. The quantitative analysis compares the stitched position of matched image feature pairs found with the Shape Invariant Feature Tracker (SIFT) and Speeded Up Robust Features (SURF) algorithms and shows the stitching to be accurate to within 0.3°.

The noise sensitivity of the APCCS is assessed via the generation of synthetic light source centres and robot poses. The data is realistically created for a hypothetical camera pair via the corruption of ideal data using seven noise sources emulating the robot movement, camera mounting and image processing errors. The calibration and resulting stitching accuracies are shown to be largely independent of the noise magnitudes in the operational ranges tested. The APCCS is thus found to be robust to noise.

The APCCS is shown to meet all its requirements by determining a novel combination of calibration parameters for cameras regardless of their properties in a noise resilient manner.

With thanks to:

Dr Chantal Babb

for her support, advice, warmth, motivation and excellent example.

# Acknowledgements

I would like to thank the Council for Scientific and Industrial Research, specifically the Optronic Sensor Systems competency area within the Defence, Peace, Safety and Security business unit, for their continuous support in terms of funding and access to equipment and facilities.

I would like to thank:

- Robert Jermy for many hours of capturing calibration data in the lab.
- Dean Aucamp for designing the embedded electronics for the both idealised helmet tracker and the two camera arrays.
- Ipeleng Mathubula for designing the required mechanics for the idealised helmet, to mount Light Emitting Diode (LED)s on the robot arm and to mate the robot to the optical table.
- Yazeed Schloss for designing the mechanics of the demonstration prototype camera systems.
- Bernardt Duvenhage and Asheer Bachoo for discussions on image processing and mathematics.

# Contents

# List of Figures

# List of Tables

# Acronyms

| | |
|---|---|
| **1D** | One Dimensional |
| **2D** | Two Dimensional |
| **3D** | Three Dimensional |
| **AHP** | Analytical Hierarchy Process |
| **ANN** | Artificial Neural Network |
| **APCCS** | Automatic Photogrammetric Camera Calibration System |
| **BB** | Ball Bearing |
| **BFROST** | Binary Features from Robust Orientation Segment Tests |
| **BRIEF** | Binary Robust Independent Elementary Features |
| **BRISK** | Binary Robust Invariant Scalable Keypoints |
| **CAD** | Computer Aided Design |
| **CCD** | Charge Coupled Device |
| **CF** | Coordinate Frame |
| **CFR** | Cost Function Result |
| **COG** | Centre of Gravity |
| **CPU** | Central Processing Unit |
| **CZ** | Chen and Zhang |
| **DFOV** | Diagonal Field of View |
| **DOF** | Degrees of Freedom |
| **DU** | Distorted to Undistorted |
| **FOV** | Field of View |
| **GPU** | Graphics Processing Unit |
| **HFOV** | Horizontal Field of View |
| **HL** | Hidden Layer |

| | |
|---|---|
| **HSV** | Hue Saturation Value |
| **HTS** | Helmet Tracker System |
| **IO** | Input/Output |
| **IR** | Infrared |
| **LCD** | Liquid Crystal Display |
| **LED** | Light Emitting Diode |
| **LFA** | Leap Frog Algorithm |
| **LM** | Luchesse and Mira |
| **LMA** | Levenberg-Marquardt Algorithm |
| **LSE** | Least Square Error |
| **LWIR** | Long Wave Infrared |
| **MTF** | Modulation Transfer Function |
| **PC** | Personal Computer |
| **PDF** | Probability Density Function |
| **PMJ** | Precision Mechanical Jig |
| **RADAR** | Radio Detection and Ranging |
| **RGB** | Red Green Blue |
| **RMS** | Root Mean Square |
| **SEP** | Separation of the Extrinsic Parameters |
| **SIFT** | Shape Invariant Feature Tracker |
| **SURF** | Speeded Up Robust Features |
| **TBHO** | Thermal Based Hue Offset |
| **TTGS** | Thermal Tinged Gray Scale |
| **UD** | Undistorted to Distorted |
| **UV** | Unit Vector |
| **VFOV** | Vertical Field of View |
| **w.r.t.** | with respect to |
| **ZAR** | South African Rand |

# Nomenclature

**Coordinate Frame (CF)**—A set of orthogonal axes obeying the right hand rule. CFs are used as reference systems allowing the position of a point or pose of a second CF to be meaningfully expressed. Section 3.1.1 defines the camera CF.

**Field of View (FOV)**—Angle subtended by the image plane of the camera. Accepted to mean the Horizontal Field of View (HFOV) if not specified as the Vertical Field of View (VFOV) or Diagonal Field of View (DFOV).

**Orientation**—The three dimensional angular rotation of one CF relative to another CF. Expressed as a triplet of Euler rotation angles but used in $3 \times 3$ rotation matrix format for calculation purposes. Sections 3.1.2 and 3.3 define the notation and mathematical operations of orientations.

**Point**—An infinitesimally small space which has position but not orientation. A point exists independently of any CF. Points are thus not synonymous with a vector describing their position relative to a CF. The same point's position could be expressed relative to a second CF with a second vector.

**Pose**—The complete 6 Degrees of Freedom (DOF) description of one CF relative to another CF. A pose thus consists of both orientation and position.

**Position**—The three dimensional translational offset of one point relative to another point in a specified CF.

**Vector**—A delta position between two points or CFs. A vector is expressed in a specified CF. Expressing the same vector in a different CF does not change its magnitude, merely the relative magnitude of its projection onto the orthogonal axes of the new CF. Section 3.1.2 defines the notation used for vectors and Sections 3.3 and 3.4 detail their mathematical manipulation.

# Chapter 1

# Introduction

This chapter presents the problem: the automatic determination of camera photogrammetric calibration parameters. After the problem definition, the primary research questions are presented. Finally, the outline of this thesis is provided.

## 1.1 Problem definition

Cameras are an increasingly prevalent sensor in modern society. For scientific purposes they have multiple benefits as listed below:

1. Cameras are sensitive in spectra beyond merely the visible portion of the electromagnetic spectrum.
2. Cameras can operate in both extremely low and extremely high light levels.
3. Cameras can see to the limit determined by atmospheric conditions and earth curvature, allowing remote measurements to be made.
4. Cameras can have high magnification allowing the study of minute details.
5. Cameras can operate/record continuously allowing for reliable measurements for extended periods.
6. Camera recordings can be stored indefinitely and retrieved with perfect recall for historical comparisons.
7. Cameras can operate in environmental conditions lethal to humans.

8. Cameras have no political, cultural, or religious bias.
9. Cameras can simultaneously observe several items within their Field of View (FOV).

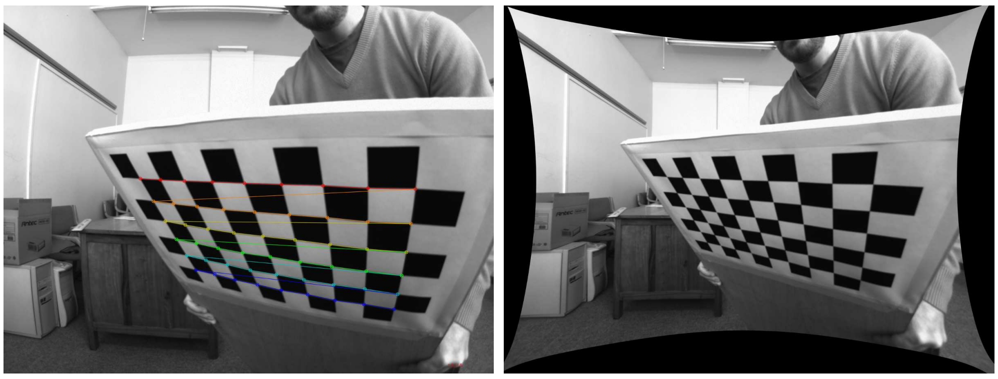
In order to use a camera an accurate and repeatable measurement instrument it is necessary to calibrate it. Camera calibration is a set of measurements performed on a camera such that it can subsequently be used to make further measurements.

### 1.1.1 Camera calibration

Camera calibration is a core technology that is used in almost all electro-optical equipment. There are two kinds of camera calibration:

1. **Radiometric calibration** is the process of equalising or characterising the electrical response of the pixels, so that one can determine the optical energy of the light that is falling on each pixel. Thereafter the image can be equalised to give smooth outputs even if there is significant vignetting caused by the lens. This is typically what is done on Infrared (IR) cameras so as to obtain IR signatures of aircraft, missiles and items of military interest. Willers' extensive work [1] on radiometry can be consulted for further information on radiometric calibration and systems.

2. **Photogrammetric calibration** encompasses all the geometric properties of the imaging system. It entails the determination of all the parameters such that an image coordinate can be converted into a vector in a chosen reference Coordinate Frame (CF) and a point in the reference CF can be converted into an image coordinate. This allows the direction to objects to be determined. In certain circumstances this can then allow the size, distance and orientation of objects viewed by the camera/s to be determined.



(a) Checker board input image. (b) Distortion corrected image.

Figure 1.1: OpenCV checkerboard based camera calibration.

### 1.1.2 Photogrammetric camera calibration

Photogrammetric camera calibration is useful for many purposes ranging from autonomous navigation [2] of air, ground [3] and underwater vehicles, to augmented reality and Three Dimensional (3D) modelling amongst others. Two applications are presented: real-time panorama stitching and optical helmet tracking.

Photogrammetry involves the determination of two sets of parameters:

1. **Intrinsic calibration** is the determination of the internal camera parameters that allow for converting back and forth between image coordinates and 3D vectors expressed in the local camera CF (Section 3.1.1). These parameters include the focal length of the lens, size and orthogonality of the pixels, lens distortion parameters (and inverse distortion parameters if desired) and the principal point.

2. **Extrinsic calibration** determines the pose of the cameras relative to each other and the reference CF in all 6 Degrees of Freedom (DOF). These DOFs are the translations along $X$, $Y$ and $Z$ axes as well as the yaw, pitch and roll angles as defined in Section 3.1.1.

In many of the methods surveyed in literature (Section 2.3) camera calibration is performed manually using a checkerboard and a toolkit such as OpenCV [4] to process images of the checkerboard. Figure 1.1 shows typical results. Such a technique has several limitations:

1. The spectrum at which checkerboards are visible (or custom boards are manufactured) is limited. For instance, a checkerboard in the Long Wave Infrared (LWIR) would require adjacent tiles to have a step edge in temperatures, which is impractical.
2. The calibration repeatability is severely compromised by the requirement to manually present the board to the camera at different orientations in the camera's FOV.
3. The quality of the data can be affected by blur caused by motion of the checkerboard during the camera's image acquisition phase.
4. The throughput of such a process may be low due to the human involvement.
5. It has been shown to have errors up to 20% larger when compared to high order calibrations [5] which are presented later in this work.
6. Cameras in a single system but with differing sensitivity spectra cannot be calibrated.

### 1.1.3 Automated camera calibration

It is clearly desirable to mitigate some of the effects of checkerboard based camera calibration. One possible solution is to either use electronic checkerboards or to create an Light Emitting Diode (LED) array or array of similar reference marks. This method too has some limitations:

1. LED arrays are time consuming and expensive to create and to calibrate. The wiring and individual control of several hundred LEDs is not a trivial task. Measuring the 3D position of each LED (which is required) typically involves deploying and aligning several theodolites, which then need to each be manually aimed at each LED in order to triangulate its position. To provide a single point of reference, it cost approximately 150k South

African Rand (ZAR) to produce a custom 500 LED array in 2006 at my former employer and the ABB IRB120 robotic arm, (which is an expensive high precision robotic arm, see Table 7.3) used in this work cost 220k ZAR in 2012. Taking the official South African Government's 6% inflation figure as guideline, the LED array and robot arm are priced equivalently, but the former is a standard item with less lead time to procure/produce and is more versatile as discussed below.

2. Changing the FOV of the cameras may require the creation of a new checkerboard or LED array.
3. The system is still not easily able to handle cameras with different sensitivity spectra.
4. There is limited ability to trade off calibration speed versus number of data points captured.
5. Different camera resolutions can induce problems in differentiating between adjacent densely packed LEDs or checker intersections.

An ideal Automatic Photogrammetric Camera Calibration System (APCCS) would then have the following attributes:

1. The calibration process is automatic.
2. The data acquisition is repeatable.
3. The system caters for cameras of different sensitivity spectra.
4. The system caters for cameras with different FOVs.
5. The system caters for cameras with different pixel resolutions.
6. The trade off between calibration accuracy and calibration time can be altered.
7. The system must determine all the required photogrammetric parameters.
8. The system can calibrate any number of cameras.

It was decided to solve this problem by using a precision mechanical device to position a configurable light source at known positions in the FOV of the camera being calibrated. This research investigates whether a system using a robotic arm as the mechanical device can achieve the desired attributes stated above. The small, 6 axis, 25 kg, 10 $\mu$m accurate ABB IRB120 robotic arm with a 3 kg payload was identified for use to develop and test the APCCS.

## 1.2 Research questions

The problem definition and context resulted in the following research questions:

1. Can an automated photogrammetric camera calibration system meeting the criteria listed in Section 1.1.3 be created using a robotic arm?
2. Are the calibration parameters produced by such a system suitable for real world applications?
3. Is a robotic arm based photogrammetric system sufficiently robust to measurement and movement noise?

## 1.3 Contribution of this work

This work made several novel contributions to the body of scientific knowledge. When the project description was formulated in 2010 there was no literature available on using a robotic arm for photogrammetric camera calibration. However in 2011 Peters et al. [6] patented a system for the calibration of stereo cameras.

The system of Peters et al. did not however fulfil all stated requirements for an APCCS in Section 1.1.3. Their system cannot calibrate only one camera and not all the required parameters were determined. Specifically systems calibrated by the system of Peters et al. cannot perform absolute ranging as normalised units and focal lengths are used.

The system proposed in this thesis, which fulfils all the stated APCCS requirements, is thus novel. So too is the application of the system to photogrammetric stitching and the error sensitivity study of the system.

This work has resulted in two papers [7, 8] and an international patent [9] regarding the APCCS. A further three papers on camera photogrammetric calibration [5, 10, 11] and three more on improved photogrammetric stitching [12], multispectral registration [13] and multispectral fusion [14] were also published. The author of this thesis was the lead author of all the papers with the exception of the multispectral registration paper [13] which had a stronger emphasis on feature based processing rather than photogrammetry.

## 1.4 Thesis overview

The primary aim of this research is to address the problem of photogrammetric camera calibration. This chapter showed that current methods for repeatable photogrammetric calibration of cameras with varying resolutions, FOVs and sensitivity spectrums were lacking. A solution to this problem using a robotic arm was presented. The complete photogrammetric calibration system, the APCCS and its ideal characteristics were discussed. Research questions regarding the feasibility, real world applicability and noise robustness of such a system were presented.

A thorough literature review is undertaken in Chapter 2. The emphasis is placed on the characteristics and suitability of methods for the APPCS, with the mathematics of the most suitable methods left to Chapter 3. The diverse fields of numerical optimisation (Section 2.2), camera calibration (Section 2.3), camera pose estimation (Section 2.4), and photogrammetric stitching (Section 2.5) are explored in the literature review.

Chapter 3 provides the mathematical building blocks used in the APCCS. The notation and axes definition is given in Section 3.1. Global and local numerical optimisation algorithms are explained in Section 3.2. Section 3.3 presents the required background on 3D rotations, including the usage of the non-standard angles output by ABB IRB120 robot arm. The chapter ends with Section 3.4 providing the routines for processing LED images, converting between Two Dimensional (2D) image space and 3D coordinate frames and back, as well as 3D trigonometric algorithms to find line intersections and the camera pose via observation of a known tetrahedron.

Chapter 4 presents the initial work done to determine the direction followed to develop the APCCS. Section 4.1 investigates whether explicit models or black-box models as well as whether polar or Cartesian models yield superior calibration results. It is shown that explicit polar models yield the best results. Two papers [10, 11] were published on the use of Artificial Neural Network (ANN) to perform this black box modelling. Section 4.2 studies which polar models and calibration patterns provide the best results in the context of monocular 3D spatial measurements. It was found that high order Brown models (when stably fitted using appropriate numerical techniques) provide the best real-world performance. These results were published [5].

Chapter 5 explains the algorithms for the robotic arm based camera calibration and contains the primary novelty in this research. The novelty stems from the unprecedented combination of capabilities of the APCCS, the ability to calibrate a single camera, and the mathematics of several of the calibration algorithms. A paper [7] providing an overview of the system and its application to stitching and helmet tracking was published.

Sections 5.1 and 5.2 provide a physical description and high level overview of the operation of the APCCS, respectively. The calibration dependencies and correct order of calibration are also discussed.

Section 5.4 describes the system calibrations required before the APCCS is ready to calibrate cameras. The calibration of the removable Precision Mechanical Jig (PMJ) (which is the only item required to be calibrated externally) is also described. Since the APCCS is designed to be easily adapted to new cameras with different spectrums and FOVs, the ability to determine the position of new light sources mounted on the robot arm is required. This determination uses two mounts of the PMJ to form a stereo pair. Two sequences of robot movements are observed from each mount. The first sequence has a constant orientation of the end effector but varies the translation to allow the alignment of the stereo pair's CF and the robot arm's CF to be determined. The second sequence has a constant translation but varying orientation of the end effector and allows the light source's spatial offset to be determined using the measured stereo pair to robot orientation.

Section 5.5 provides the details of the camera calibration routines which are the core of this research. For lens calibration in the Distorted to Undistorted (DU) direction, the end effector moves the LED through a sequence of discrete points forming a large 2D grid. The computer then determines the LED position in the camera's image and records this position together with the end effector pose. The computer processes these results to find the best fit parameters for a high order Brown model which yields corrected LED image positions that are maximally collinear. These resultant corrected image points are then processed to determine a set of Undistorted to Distorted (UD) parameters by finding the high order Brown parameters that best map the corrected image positions back to the original captured positions.

With the DU and UD calibrations complete, the focal length is determined by a novel method of optimising a hypothesised focal length to result in the tightest possible cluster of relative camera poses. These poses are analytically calculated from observing the same set of tetrahedrons from two different mounting points. This is possible because the robot moves the LED to the vertices of the tetrahedrons in a repeatable manner. This means that the vertices are known in a common CF (the robot arm CF) and so the calculated camera pose from any tetrahedron should be identical.

The APCCS allows the extrinsic position of the camera to be separated into two poses to facilitate the easy replacement of cameras in a deployed end-user system without repeating the calibration of the entire end-user system. This separation is performed by the camera observing the same sequence of points (which formed a 3D grid) from each of the purposefully misaligned mounts (whose relative poses are known) on the PMJ. The system determines the pose of the the camera with respect to (w.r.t.) the mounting interface and the pose of the PMJ w.r.t. the robot. This simultaneous determination of the two poses is novel and allows pose of the PMJ w.r.t. the robot arm to not be known a priori. This means that the PMJ can be moved on the table to the location most suitable for the current camera being calibrated.

Chapter 5 ends with a description of the measurement of the mount pose w.r.t. the robot arm. This entails a camera, whose pose w.r.t. its mounting interface is already known, to observe a single sequence of LED positions constituting a 3D grid. From the determined LED image coordinates and end effector poses, the pose of the camera w.r.t. the robot is determined. From this 'total' camera pose, the pose of the camera w.r.t. the mount is 'subtracted' to yield the desired pose of the mount w.r.t. the robot.

Chapter 6 investigates the applicability of the APCCS outputs to real world applications. Photogrammetric stitching is extensively investigated in Section 6.1. The mathematics of the stitching process are optimised (Section 6.1.1) for quicker throughput and decreased depth sensitivity of the stitch and were subsequently published [12]. In order to verify that the APCCS can calibrate cameras of different spectrums, arrays of cameras consisting of both visual and LWIR cameras are evaluated. This requires a sensible method of simultaneously displaying visual and LWIR images, on a visual spectrum only display. Section 6.1.2) present a novel application of (and extension to) the Analytical Hierarchy Process (AHP) that is used to select the best fusion method. The results of this fusion investigation were published [14]. A quantitative metric to assess the accuracy of the stitching based on projecting matched Shape Invariant Feature Tracker (SIFT) and Speeded Up Robust Features (SURF) image features in the overlap regions onto the stitch surface is presented in Section 6.1.3. Subjective pictorial results as well as quantified stitching errors for two camera systems (both One Dimensional (1D) and 2D camera arrays) are then presented in Section 6.1.4 and found to be acceptable. This shows that the APCCS outputs are both accurate and suitable for real-world use. The photogrammetric stitching is found to work reliably and correctly even in situations of poor focus, lens flares and brightness variations resulting from images caught in uncontrolled outdoor conditions. The SIFT and SURF features are far less robust to these aberrations, highlighting the unsuitability of feature based stitching for outdoor surveillance.

Chapter 6 presents a brief overview of the results of using the APCCS outputs for optical helmet tracking in Section 6.2. Details of a working prototype optical helmet tracking system are presented. It is seen that the laboratory tracker provides helmet pose measurements whose accuracies are similar to systems currently in operational use. This further highlights the suitability of the APCCS outputs to real world applications.

Chapter 7 investigates the sensitivity of the APCCS outputs to noise. Both the inability of the end effector to move to the perfect pose or to perfectly report the achieved pose are modelled. In addition, the non-perfect repeatability of the camera mounting interface and the error in LED localisation in the image are modelled. Section 7.1 shows how these errors are used to synthesise realistic noisy robot poses and LED image positions for all the calibrations of Chapter 5. The measurements synthesised are based on a physical equipment configuration representative of that used to calibrate the cameras for the stitching accuracy assessment of Section 6.1. These calibrations are used to determine the error between the ideal image coordinate and the 'as calibrated' image coordinate of the representative hypothetical two camera stitched system. This evaluation covers the entire FOV of the cameras, not merely their overlap and is independent of scene depth effects.

Section 7.2 describes the design of the simulation experiment. Multiple samples of noise levels ranging from zero to beyond the expected system values are used to synthesise the LED image coordinates and the robot pose data. This data is used to calibrate the hypothetical camera pair and test the stitch accuracy. Section 7.3 provides the result of the simulation runs. The stitching accuracy is plotted versus noise levels and the correlation between noises sources, calibration accuracies and stitching accuracies is determined. It is seen that neither the calibration nor the stitching accuracy deteriorate significantly over the noise ranges tested. The system is thus deemed resilient to robot movement inaccuracies, camera mounting repeatability, and LED localisation noise. A simplified version [8] of this simulation study was published.

Chapter 8 is the conclusion. Section 8.1 provides a review of the results and decisions taken throughout the research. The research questions are then revisited and answered, in light of the results achieved, in Section 8.2. Section 8.3 presents the key findings of the development and testing of the APCCS. Finally, Section 8.4 discusses further work that could be undertaken to continue this research.

# Chapter 2

# Literature survey

This chapter contains an overview of the relevant research techniques and methods and their development. The methods and techniques are compared and their relative benefits and best application are discussed. The mathematical details are intentionally omitted from this chapter to better concentrate on the characteristics of each algorithm. Chapter 3 contains the mathematical details of those methods and techniques used in subsequent chapters.

Section 2.1 discusses previous research by the author that is relevant to this thesis.

Section 2.2 provides an overview of numerical optimisation methods. These methods are used extensively in Chapter 5 for camera calibration. Each calibration is posed as an optimisation algorithm by finding a cost function that, when minimised, results in the desired calibration parameters.

Section 2.3 provides an overview of lens distortion calibration techniques. This includes aspects of how the distortion is modelled and how the data for the calibration data are obtained. This section thus is pivotal in the development of the APCCS.

Section 2.4 discusses how the pose of the camera w.r.t. the object it's viewing (or vice versa) is determined. Pose determination is used in the camera focal length determination (Section 5.5.3) and is the primary mechanism of the helmet tracker example application (Section 6.2).

Section 2.5 discusses stitching from a photogrammetry perspective. Stitching is used extensively in later chapters both as an example application (Section 6.1) and as a real-world accuracy measure of the error sensitivity analysis of the APCCS (Chapter 7).

## 2.1 Previous work by the author

The author has prior experience in photogrammetric camera calibration. His master's dissertation [15] entailed the precision modelling and correction of lens distortion and the real time correction and resulted in two publications [16, 17]. Those publications serve as the basis of the lens distortion calibration aspects of this research. Prior to commencing his doctoral work the author published a paper on the fundamentals of photogrammetric image stitching [18]. The photogrammetric stitching and fusion presented in this thesos are improvements upon that original paper.

Some aspects of these previous works, where relevant, are included in the rest of this literature study and in Chapter 3, which provides the mathematical background. These sections, while crucial to this work, do not constitute the novel components.

## 2.2 Non-linear numerical optimisation

Non-linear numerical optimisation is the process whereby the parameter set that yields the optimal output from a function is sought. The function being optimised could be known in explicit mathematical form or it could be the output of a black-box process. These latter are when no knowledge of the inner workings are known, only the output resulting from a set of input parameters can be observed. A significant delay may be associated with generating outputs. Such delays may be due to a large amount of processing required in a simulation or physical limitations such as chemical reaction rates. Ergo it is typically desirable to test the fewest number of parameter sets to determine the optimal parameter set. Thus brute force searching is often unfeasible, especially as the number of parameters increase.

For this work only continuous smoothly differentiable functions need be considered. There are two kinds of optimisation for such functions: local optimisation (Section 2.2.1) and global optimisation (Section 2.2.2).

### 2.2.1 Local optimisation

Local optimisation attempts to find the best solution in the vicinity of a given starting position. This solution is not guaranteed to be globally optimal due to its dependence on the starting point provided. Local optimisation techniques typically determine a search direction which reduces the problem to a one dimensional problem and finds the minimum in that direction. Thereafter a second search direction is chosen and the minimum in that direction is found. This process is repeated until a minimum of the multidimensional function has been found. The iteration ends either when the magnitude of the gradient is deemed sufficiently close to zero, the difference between successive iterations drops below a stipulated threshold or a maximum number of iterations is exceeded.

Both the method of choosing the search direction and the method of finding the one dimensional minimum vary between local optimisation methods. Snyman [19] discusses the history of optimisation techniques. Burden and Faires [20] provide both an overview of minimisation as well as several numerical recipes, some of which were used in this work. Figure 2.1 shows an example of two methods finding the minimum of an elliptical valley.

Steepest descent (Section 3.2.2.2) is the simplest method to choose a search direction. At each point where a direction is required the negative of the gradient vector (which is the direction of steepest ascent) is calculated. The side effect of this method is that each subsequent search direction is orthogonal to the last.

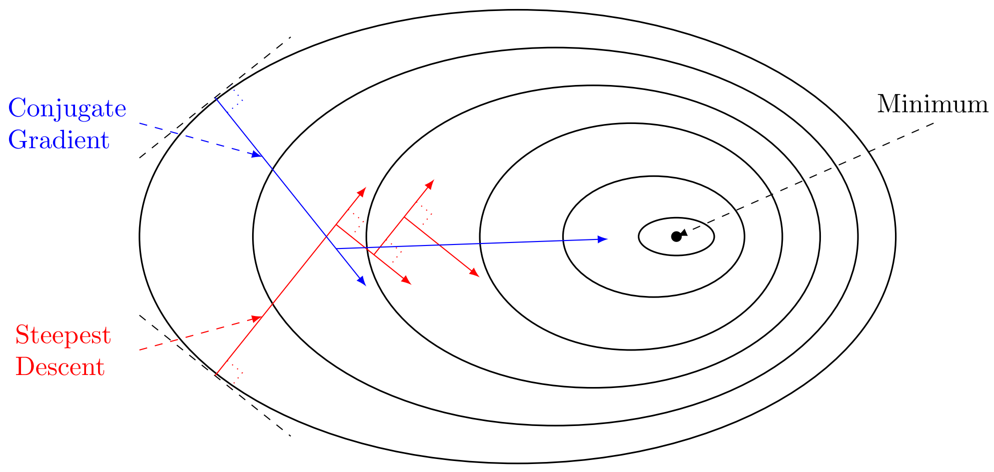


Figure 2.1: Gradient descent methods in an elliptical valley.

This can result in subsequent iterations zigzagging down to the minimum. This means that steepest descent can take longer to converge to a solution (theoretically infinite) but is guaranteed to always find a solution. Steepest descent does not specify how to find the uni-dimensional minimum at each iteration. Typically either a brute force search or a bracketing method as described in Section 3.2.2.1 is used.

An improvement to steepest descent is to use the previous search directions to better choose the next search direction. This can dramatically decrease the number of iterations, as illustrated in Figure 2.1. These are called conjugate gradient methods, the most popular of which are the Fletcher-Reeves [21] and Polak-Ribière [22] variants. The Fletcher-Reeves method is used in this work and is detailed in Section 3.2.2.3. Fletcher-Reeves is guaranteed not only to converge, but to converge quicker than steepest descent. It is guaranteed to converge in only N iterations for quadratic functions of N variables. Fletcher-Reeves also does not specify a uni-dimensional search technique.

Newton's original method was used to find the roots of a one dimensional function. At a given starting point the gradient is extended to where it crosses the $X$ axis. This means that no search is required, as the next guess of the zero is found directly based on the local gradient information and the current height of the function. To extend this method to find a minimum of a function, the roots of its first derivative are found. Similarly, the gradient vector can be used for functions of more than one variable.

Newton's method does not require a linear search as the next estimate of the minimum is found based on the local Hessian matrix (the second order derivatives) and function value. This means that Newton's method can have extremely rapid convergence. However, the calculation of the next step assumes that the local gradient is a good estimate of the entire problem space. This assumption is less true the more rapidly the function changes and the further away the starting point is from the nearest minimum. Specifically there is an assumption that the local area is quadratic (in $N$ dimensions) when each iteration is performed. If Newton's method converges it does so with few iterations. However each iteration requires significantly more computational effort in order to calculate the Hessian matrix. If the Hessian matrix has to be estimated from function evaluations, a total of $2N^2 + 2N$ function evaluations are required if the Hessian matrix's symmetry is exploited. Newtonian iteration can thus be undesirable if the function evaluations take a long time, or the cost function is unlikely to be approximately quadratic.

The Levenberg-Marquardt Algorithm (LMA) was independently discovered by Levenberg [23] in 1944 and formulated by Marquardt [24] in 1963. LMA is a damped Newtonian method: an interpolation factor biases each iteration closer to either the Newtonian step or towards a steepest descent style gradient step if the iteration is starting to become unstable and diverge. LMA is intended to be the ideal cross between the fast convergence of Newton's method and the guaranteed convergence of steepest descent. For this reason LMA is an extremely widely used algorithm and is the de facto method used in many standard optimisation suites including Mathworks' Matlab [25]. Being a Newtonian method, LMA also requires the derivation or estimation of the Hessian matrix with the associated computational implications as already discussed. A side effect of biasing the iteration towards steepest descent is that the step size decreases. This can mean that the iteration will falsely converge either because the step size drops below the threshold or because the maximum number of iterations is rapidly depleted by the small steps. Marquardt [24] stated that false convergence will happen in problems with highly correlated parameters.

The final and newest method considered is Snyman's Leapfrog method [26] so named after the iterative numerical integration method it uses. Leapfrog simulates the movement of a charged particle in an $N$ dimensional electric field, details are provided in Section 3.2.2.4. The particle experiences accelerations induced by the gradient of the function being optimised. The particle gains both velocity and momentum from these accelerations, meaning that it can go over small humps and finds not merely the nearest local minimum, but the nearest 'low local' minimum to the starting point. Leapfrog does not make any explicit line searches and is guaranteed to converge to a minimum. Leapfrog takes longer to converge than conjugate gradient methods [19] but is robust to noise and similar perturbations.

Since all the optimisations in this work are performed only once per calibration (Chapter 5), Leapfrog was chosen for its superior robustness and ability to find better local minimums.

### 2.2.2 Global optimisation

Local optimisation (Section 3.2.2) only finds the best solution in the vicinity of a provided starting set of parameters. In contrast, global optimisation is the search for the set of input parameters that result in the best solution over the entire feasible region of the parameter space. Neumaier [27] provides an introduction to global optimisation. He categorises the available algorithms by the degree of rigour with which they find the global optimum:

1. **Incomplete methods** use heuristic based search techniques and have no safeguards to avoid getting stuck at a local minimum.
2. **Asymptotically complete methods** will find the global optimal solution given infinite run time but have no means of identifying when they have reached the global optimum.
3. **Complete methods** will find the global minimum (assuming exact calculations) given indefinite run time but are able to discern when the global

optimum has been found to within specified bounds.

4. **Rigorous methods** will find the global minimum to within specified tolerances with certainty even in the presence of rounding errors, except for degenerate cases.

Neumaier further states that the first two categories are often the most useful for large scale problems that have only black-box evaluations which do not provide any global information. Within this category prevalent techniques are simulated annealing [28] and multi-agent methods (discussed below). Simulated annealing seeks to emulate how metals cool into crystalline structures with the lowest possible energy state. Multi-agent techniques use heuristics to improve upon the basic method of multiple random starting points for local optimisation.

Multi-agent searches differ primarily in how the search points in the next iteration are derived from the current search points, based on the relative optimality of each search point. Prevalent techniques in this category are particle swarm optimisation [29], differential evolution [30], and genetic algorithms [31, 32].

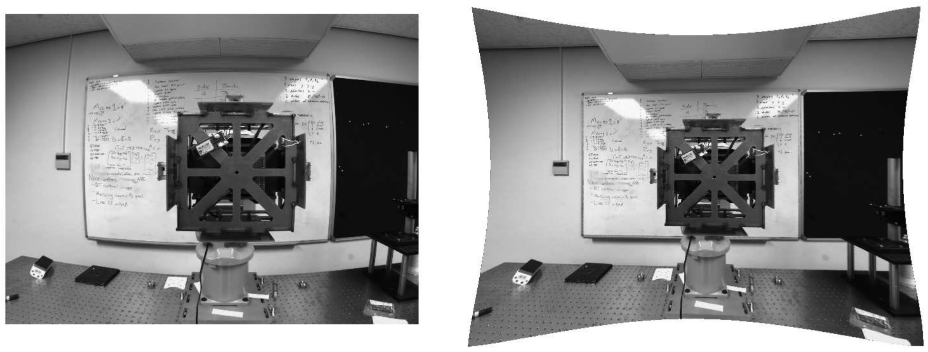
Kennedy and Eberhart [29] created the particle swarm optimiser by adapting a simplified simulation of the movement of a flock birds. Each bird (which originally has a random position and velocity) knows both the best position it has personally encountered and the best position the flock has encountered. At each iteration each birds' velocity is simultaneously adjusted by a random scaling of the vector directing it towards its personal best and second independent randomly scaled vector towards the flock's best position. These random scale factors are in the range of zero to one. After all the velocity adjustments are made, each bird's position and potentially each bird's and the flock's personal best positions are updated. Kennedy and Eberhar removed velocity matching amongst neighbouring birds, this results in the flock behaviour being less synchronised. Hence, the terms 'particle' and 'swarm' are used in the vernacular rather than 'bird' and 'flock' respectively.

Storn and Price [30] introduced differential evolution in 1997. Each new iteration of solution sets is created from the previous iteration. A candidate solution for the new iteration is first created by taking a linear combination of the parameters from three current solutions. Each parameter of this solution is then randomly swapped with that of a fourth current solution. This new candidate is then compared to the fourth solution and the better solution propagated to the next generation. This is repeated until either a suitable minimum is found or the maximum number of generations is exceeded.

Genetic Algorithms [31, 32] are based on the theory of evolution. Each solution is termed an 'individual' with the parameters of each solution being coded into a 'chromosome'. The quality of each solution presented by an individual is evaluated and then normalised. The next generation is created by choosing, in a fitness biased manner, two individuals and then either cloning them or combining their 'chromosomes' to create two new individuals. New individuals are repeatedly created until the next generation of individuals is fully populated. Successive generations are created until either a suitably fit individual is found or the maximum number of generations has been exceeded. The number of individuals per generation; the number of generations; the fitness biased method of selecting individuals for procreation; and the manner in which offspring are created from their parents all need to be defined for the specific problem being addressed. Genetic algorithms are used in this research. Further details on the implementation can be found in Section 3.2.1.

## 2.3 Lens distortion correction

Most optical lenses induce some amount of distortion in the images they create, that is: the resultant images differ from what one would get with an ideal pinhole camera in terms of the projection. In addition to the obvious aesthetic degradation, this has severe consequences for any measurements calculated from the image or for applications such as machine vision, photogrammetric stitching, product defect detection, remote sensing and motion tracking. While it is possible to correct all or some of the distortion optically, this increases the size, mass and price of the lenses considerably and so the correction is frequently done in software. Figure 2.2 shows an example of a distorted image (Figure 2.2(a)) and



(a) Distorted image. (b) Undistorted image.

Figure 2.2: Matching distorted and undistorted images.

its corresponding undistorted image (Figure 2.2(b)). The corrected image was created using the APCCS' DU and UD outputs.

The distorted image in Figure 2.2(a) is an example of barrel (or mild fish eye) distortion. Barrel distortion is typical of many short focal length lenses. Points are moved towards to the centre of distortion. The distortion centre and image centre are normally close but not identical. The further away the points are from the centre the more they are moved inwards. This results in straight lines tangential to image radii appearing curved inwards. The curvature is due the extreme points of the lines being further from the centre of distortion than the middle of the line, and hence these extreme points are pulled closer to the distortion centre. This gives rise to the name barrel distortion as rectangles become distorted into barrel-like shapes. The correction of these images yields the classical bow-tie outline as evidenced in Figure 2.2(b). The opposite of barrel distortion is pincushion distortion where points are pushed outwards. This is much less common and typically is only found in low-end telephoto lenses.

The causes of the lens distortion can be intentional, i.e. lens distortion is allowed in the optical design as it can lead to smaller, lighter lenses with fewer optical elements and, crucially, fewer aspherical elements (which can be expensive). Unintentional causes for lens distortion include manufacturing tolerances of the lenses and the mechanical assemblies that house them, variations in the refractive index of elements, and non-planarity or non-orthogonality of the image chip relative to the principal axis of the lens.

The first person to correctly explain the distortion observed from spherical element lenses was Conrady [33] in 1919. Conrady's model was based on the analytical ray tracing of light through non-perfectly aligned lens elements. This work lay largely ignored until Brown [34, 35] demonstrated both its applicability and a practical reformulation of the algorithm together with a means to determine the parameters. This became known as the 'plumb line' method. Brown's equation is given in Equation 3.25. This gave rise to the polar parametric models described in the following section. A more complete description of the history of lens distortion can be found in the works of Clark and Fryer [36] and de Villiers [15].

### 2.3.1 Polar parametric methods

Brown's plumb line method [35] is the basis for the majority of current lens distortion correction techniques [15, 16]. De Villiers [15] provides a survey of parametric lens distortion techniques in literature and lists the number of radial and tangential terms of Brown's model (Equation 3.25) used by each technique. Figure 2.3 depicts the radial and tangential distortion components in a greatly exaggerated scale.

Further examples of prevalent polar radial distortion models in literature include the widely cited albeit disproven [16] work of Tsai [37]. Zhang [38] showed that printed calibration patterns can work well and Stein [39] pioneered epipolar geometry of free images.

De Villiers et al. [16] showed that by using the appropriate numerical optimisation techniques (including appropriate parameter scaling [19]), not only could more than 2 or 3 parameters be stably fitted to Brown's model but that the extra parameters significantly decreased the residual distortion error by up to 96%. De Villiers et al. [17] further went on to show that by removing the implicit radial symmetry assumption the residual error could be further decreased by

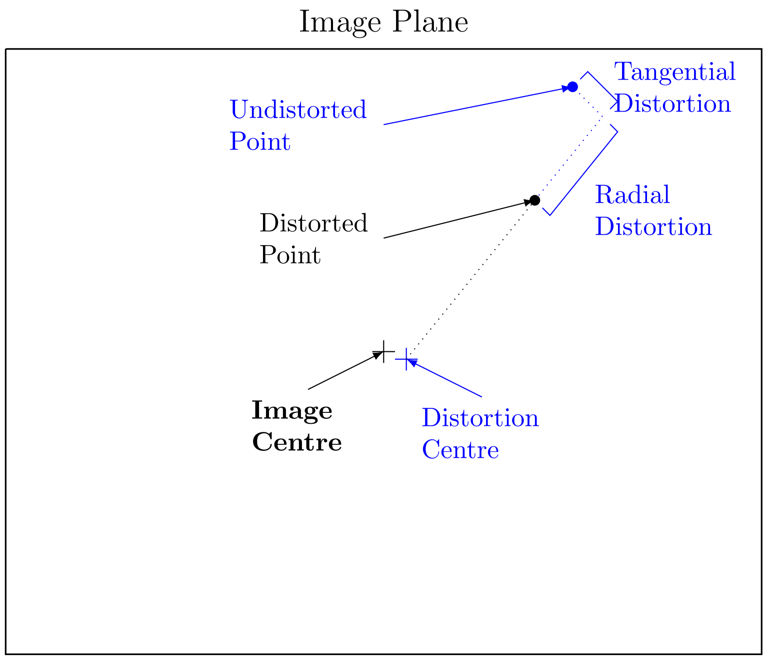


Figure 2.3: Brown distortion modes.

5%, even for the Brown models with the most parameters.

The plumb line method is based on the maxim that straight lines in the world should project onto straight lines in the image. Thus, data points known to be collinear in world space are captured and identified in the distorted image. The chosen number of parameters from the chosen distortion model are then manipulated until the distortion-corrected image points are as collinear as possible. Brown observed strings suspended in oil and held taut by suspended weights. Current methods primarily involve repeated observations of an asymmetrical checkerboard and using the checker intersections as accurately locatable points known to be collinear. Two popular software items using checkerboards are the Open Computer Vision project [4] and the California Institute of Technology's Camera Calibration Toolkit [40].

Section 2.3.3 provides more information on different calibration targets. Section 4.2 provides details on a comparison of these widely available photogrammetric calibration suites with the work of de Villiers et al. [16] for real-world triangulation and localisation applications.

### 2.3.2 Cartesian parametric methods

Not all of the lens distortion models in literature assume radial symmetry, some try to find a direct mapping between the input and output Cartesian coordinates directly. Sagawa [41] directly calculated the Unit Vector (UV) from the distorted positions. This was done by moving a physical reference placed at known positions relative to the camera and observing the image coordinates of the references to generate a coarse mapping. Pixel positions between those directly mapped have their UV created by interpolation of their nearest mapped neighbour's UV.

Claus and Fitzgibbon [42] fitted two rational functions, one correcting horizontal and one correcting vertical distortion. The numerators were independent but the denominator was common. These numerators and denominators were full second order polynomials of the horizontal and vertical ordinates with their corresponding 18 total parameters determined by epipolar geometry. Equation 2.1 illustrates:

$$\begin{aligned} h^u &= \frac{a_0h^2 + a_1hv + a_2v^2 + a_3h + a_4v + a_6}{c_0h^2 + c_1hv + c_2v^2 + c_3h + c_4v + c_6}, \text{ and} \\ v^u &= \frac{b_0h^2 + b_1hv + b_2v^2 + b_3h + b_4v + b_6}{c_0h^2 + c_1hv + c_2v^2 + c_3h + c_4v + c_6} \end{aligned} \tag{2.1}$$

where:

$$\begin{aligned} (h^u, v^u) &= \text{the undistorted image coordinates,} \\ (h, v) &= \text{the input distorted image coordinates,} \\ a_0 \dots a_5 &= \text{horizontal numerator correction parameters,} \\ b_0 \dots b_5 &= \text{vertical numerator correction parameters, and} \\ c_0 \dots c_5 &= \text{common denominator correction parameters.} \end{aligned}$$

Candocia [43] was interested in preserving the scale of undistorted images to aid image mosaicking. He did this by starting with Brown's model and removing the dependence of the vertical correction on the vertical ordinate, and then removing the dependence of the horizontal correction on the horizontal ordinate.

Similar investigations were performed during the investigative phase of this work. Specifically, a variety of ANNs with varying activation functions and architectures were trained via several different methods to determine their suitability to model inverse distortion. Both separate ANNs for the horizontal and vertical corrections as well as combined ANNs were evaluated. More information can be found in Section 4.1.

### 2.3.3 Capturing data for lens calibration

Regardless of whether polar or Cartesian methods are used, reliable data with which to perform the calibration are required. This typically takes the form of a regular grid of easily and accurately locatable points.

One of the most common calibration target types are checkerboards. Both OpenCV [4], and the California Institute of Technology camera calibration toolbox [40] use checkerboards. Other examples in literature include Mallon and Whelan [44] who model 'undistortion'. Harguess and Strange [45] investigates using visual checkerboards exposed to solar radiation to calibrate IR cameras, with limited success. Techniques to accurately locate the checker intersection leverage the non-perfect imaging resulting from limited lens Modulation Transfer Function (MTF) and Charge Coupled Device (CCD) spatial sampling. These techniques include the saddle point method of Lucchese and Mira [46] and Chen and Zhang's [47] Hessian matrix based method.

A similar family of calibration targets consist of a grid of non-touching squares. Each corner of each square is an accurately locatable point. Jeong [48] and Zhang [38] both make use of these grids.

A similar and very popular calibration target is arrays of circles, with the centre of each circle an accurately locatable point. Examples include the works of Claus and Fitzgibbon [42] and the rational function inverse distortion modelling work of Silven [49]. Yu [50] used a circular grid to create an embedded solution for distortion correction. Methods to locate the circles' centres range from simply calculating the centroid of the thresholded images to ellipse fitting methods such as that proposed by Redert et al. [51].

Using straight lines in the image is the classical method proposed by Brown [34, 35]. Other methods often assume that their accurately locatable discrete points are collinear in order to use the 'plumb line' methods. Examples of using straight lines in literature include the low level electronic implementation by Nijmeijer et al. [52] as well as the so-called scale preserving model of Candocia [43].

Tsatsakis et al. [53] also use a grid of lines. Tsatsakis et al. automate the calibration by changing the distance between the grid and camera to find when the grid lines correspond to the analogue camera's scan lines. Tsatsakis et al. then directly compute a mapping from the pixel domain to angular vector domain. Sagawa [41] observes structured light patterns created by a Liquid Crystal Display (LCD) that completely subtends the camera's FOV to create another line based automatic system.

Zhen et al. [54] use an array of circles on a four DOF '$XY\theta Z$' platform to automatically capture data throughout the camera's FOV. Peters et al. [6] use a robot arm to place a single LED in 3D relative to the cameras. This system is unable to calibrate cameras to provide absolute measurements or to calibrate only a single camera.

A comparison of the different calibration targets and methods to locate them accurately in the camera image was conducted in the initial investigative phases of this research (Section 4.2).

For the automation of the data acquisition a robot arm was selected similar to Peters et al. [6]. Three additional requirements were added to address the short comings of their system: the system must be able to calibrate a single camera, a calibrated camera must be able to make absolute measurements, and the system must be able to calibrate cameras in end user applications consisting of cameras sensitive to different spectra.

## 2.4 Pose estimation from observation of four points

Pose estimation is the determination of the orientation and translation of the camera relative to the scene that it observes. Equivalently the pose of the scene, or some part of the scene, relative to the camera can be determined. Typically this requires knowledge of the camera and lensing effects (i.e. the camera intrinsic parameters), the resulting knowledge of the camera's pose in the world is then called the extrinsic parameters.

In this work it is assumed that four non-planar points (i.e. a tetrahedron) in the scene are uniquely identifiable, accurately locatable in the image, and rigidly located relative to each other in space. If the translations between these points are known, then it is possible to uniquely determine the absolute pose of the camera relative to the tetrahedron in a deterministic and analytical manner. The procedure is outlined in an appendix of Fischler and Bolles' [55] seminal work on random sample consensus. Another formulation for solving this tetrahedron problem is provided by Kniep et al. [56]. The detailed mathematics of this process is discussed in Section 3.4.7.

This measurement of a tetrahedron's pose relative to the camera is non-contact and has many applications. Similar pose estimation techniques have been put to such diverse uses as:

1. Aiding the mating of the nozzle and catchment parachute for air-to-air refuelling [57].
2. Landing of probes on celestial bodies [58].
3. Maintenance of nuclear reactor components in environments unsafe for humans [59].

In this work pose estimation is used to determine a camera's optimal focal length (Section 5.5.3). It also forms the basis for the optical helmet tracking example application (Section 6.2) of the APCCS outputs.

Helmet trackers provide two primary benefits: off axis target designation for weapon systems and presentation of space stabilised symbology on the pilot's visor for navigation, targeting, and other mission objectives. The two primary variants are optical helmet trackers (as described here) and magnetic helmet trackers. These latter typically use three orthogonal coils mounted on the helmet and a uniform electric field in the cockpit. They then measure the electric currents induced in the coils by the movement of the pilot's head. This provides the change of pose of the helmet, which needs to be integrated over time to yield the absolute helmet pose.

Optical helmet trackers provide an absolute pose at each measurement and so do not require temporal integration. They are typically used in military applications. Some examples are Thales' hybrid inertial optical tracker [60, 61] and the Cobra helmet [62,63] used on the Eurofighter Typhoon and Saab Gripen aircraft.

## 2.5 Photogrammetric stitching

Photogrammetric stitching is the real-time creation of a panorama from an array of outward staring cameras. Panoramas are useful for persistent surveillance either of areas of high value or in the vicinity of strategic forces and assets, such as large ships.

In the maritime environment these systems are used to detect small craft in the immediate vicinity [64]. This is because small ships are hard to detect by Radio Detection and Ranging (RADAR) since they have a small cross section and are often made of materials such as wood, which has a low RADAR reflection coefficient. In addition, the small crafts' proximity means that their reflections may be received while the RADAR is still emitting energy, thus complicating the detection of their signal returns.

For this reason optical detection provides a complimentary solution to RADAR to augment the close-in detection of possible threats. It is possible to do this with a single rotating camera. Such a solution can generate extremely high resolution imagery and use superior cameras which are not economically feasible to replicate for a staring array. The disadvantages of a rotating system are that it has three additional single points of catastrophic failure: the camera itself, the rotation system, and the stabilisation platform on which the rotation system is mounted. Aburmad [65] provides further information on the advantages and disadvantages of different single and multi-aperture panoramic systems.

There are several advantages to photogrammetrically stitching multi-aperture staring camera array outputs:

1. Low detail scenes are stitched correctly.
2. Objects and details are correctly registered even if observed by cameras with vastly different sensitivity spectra (further explained in Chapter 6 and by Cronje and de Villiers [13]).
3. A single camera failing does not cause the misalignment of adjacent cameras.
4. The panorama generation is deterministic.

For commercial reasons, companies offering panoramic systems tend not to divulge the details of the workings of their systems. However, the following systems are examples of those appear to use photogrammetric stitching similar to that used in this work:

1. Immersive Media [66] who helped Google with their Street View project.
2. Point-Grey's Ladybug [67] series of cameras.
3. Thales Gatekeeper close-in protection system [68].

In this thesis the output parameters provided by the APCCS are used to stitch the outputs of two arrays of cameras with overlapping FOVs. Separate visible and LWIR stitches are created and the correspondence between pixels in the two stitches is determined from the known extrinsic parameters of the two camera arrays. Chapter 6 shows how the registration and fusion is performed.

# Chapter 3

# Mathematical fundamentals

This chapter details the basic mathematical procedures used as fundamental building blocks in the rest of this research. This chapter is included for completeness and reproducibility. With the exception of Section 3.1 it may be skipped in a first reading.

Section 3.1 provides the mathematical conventions and the notation used. Axis definitions, CFs, positive rotation directions, and the representation of matrices, vectors, vector elements, and scalars are all defined.

Section 3.2 explains the numerical optimisation routines used to find parameter sets which minimise a given cost function. These routines are crucial as numerical optimisation is used for every calibration performed by the APCCS. Where sufficient prior knowledge is available starting positions are created analytically, otherwise global optimisation routines (Section 3.2.1) find suitable starting points. These starting points are then refined by local optimisation routines (Section 2.2.1).

Section 3.3 defines the concept of a 3D rotation and shows how to convert between different representations of 3D rotations. This is important as there are typically 3 different CFs involved in each optimisation, with reference to Figure 3.2: the reference/robot CF, the end effector CF and the camera CF. The conversion between rotation formats and correct 3D trigonometric operations are required to ensure the correctness of the calibration routines. These routines are used throughout this text and so their individual uses are not discussed in their sections.

Section 3.4 provides the basic 3D trigonometric and image processing routines used. These routines are typically high level building blocks, all of the routines are used in at least one of the calibration routines (Chapter 5), example applications (Chapter 6), or the stitch robustness assessment (Chapter 7). As the routines' significance are not immediately apparent from their derivations, the explicit uses of each of these routines will be presented in their respective sections.

## 3.1 Mathematical conventions definition

This section defines both the CF conventions and the CFs used in the APCCS. The notation and conventions used for linear algebraic operations are also defined.

### 3.1.1 Coordinate frame definition

Figure 3.1 defines the right handed CF used.

With reference to Figure 3.1:

1. The $X$ axis is positive forward and for a camera based CF coincides with the optical axis.
2. The $Y$ axis is positive to the left and in a camera system is positive in the *negative horizontal* direction in the inverted image plane when using the standard convention of top left is the image origin.
3. The $Z$ axis is positive upwards and in a camera system is positive in the *negative vertical* direction in the inverted image plane when using the standard convention of top left is the image origin.

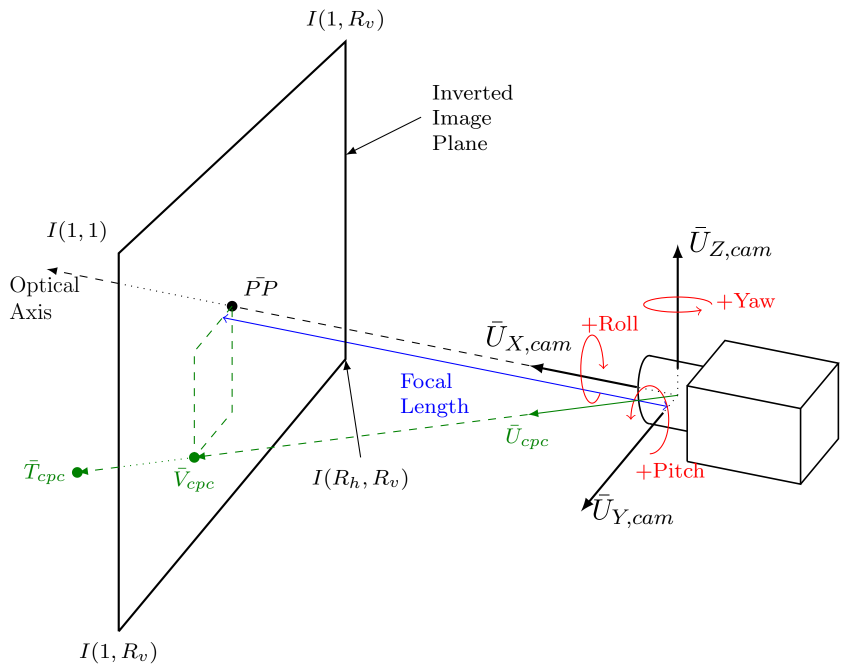


Figure 3.1: Camera coordinate frame.

4. Positive azimuth or yaw is clockwise when looking in the positive $Z$ direction.
5. Positive elevation or pitch is clockwise when looking in the positive $Y$ direction.
6. Positive roll is clockwise when looking in the positive $X$ direction.

Figure 3.2 shows the relationship between the major CFs used in this work. The robot CF acts as the primary or reference CF. The origin of the robot CF is the stationary bottom centre of the robot arm where it is attached to the optical table. The arm or end effector CF is attached to the end of the robot arm and moves with the arm as it is commanded to various poses. The robot is able to provide the pose of the arm CF w.r.t. the robot CF. The final CF is the camera photogrammetric CF shown in Figure 3.1, its pose w.r.t. the robot CF is only measurable photogrammetrically and is the focus of much of this work. All the CFs remain right handed and use the same definitions for yaw, pitch, and roll.

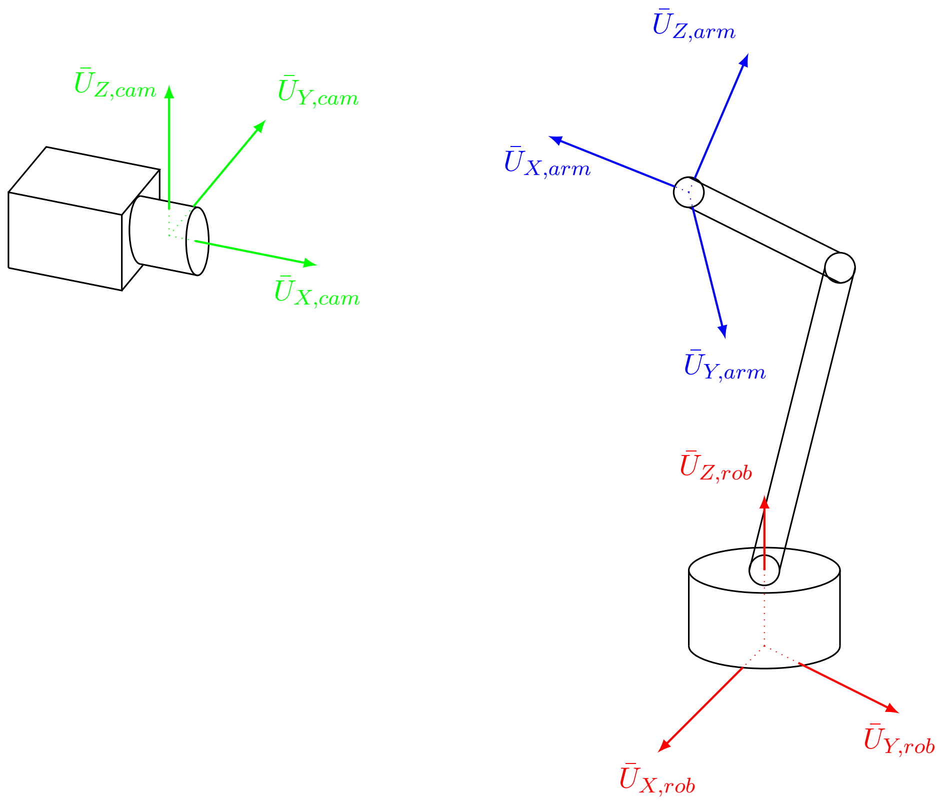


Figure 3.2: Camera, robot and arm coordinate frames.

### 3.1.2 Notation definition

The mathematical notation used in this work is as follows:

1. A 3D vector, expressed as $\bar{V}_{abc}$, is a vector from point $a$ in the direction of point $b$ expressed in terms of its projection onto CF $c$. $\bar{V}_{abc}$ is used when the magnitude of the vector is unknown or unimportant. Refer to Figure 3.3.
2. $\bar{T}_{abc}$ represents the straight line translation or displacement of point $b$ relative to point $a$ expressed in terms of its projection onto CF $c$. Refer to Figure 3.3.
3. $\bar{U}_{abc}$ is a UV pointing in the direction of point $a$ to point $b$ expressed in terms of its projection onto CF $c$. Refer to Figure 3.3.
4. $\mathbf{R}_{ab}$ is a 3x3 rotation matrix (see Section 3.3) expressing the rotation of

orthogonal CF $a$ relative to (and in terms of its projections on) orthogonal CF $b$. Refer to Figure 3.4.

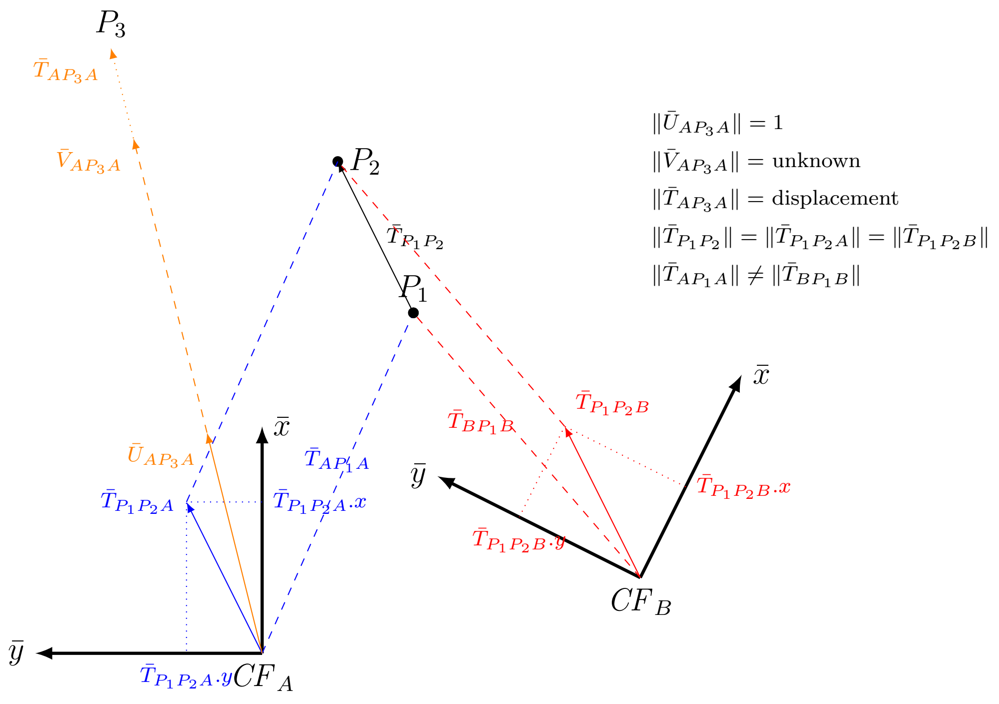


Figure 3.3: Nomenclature: vector notation.

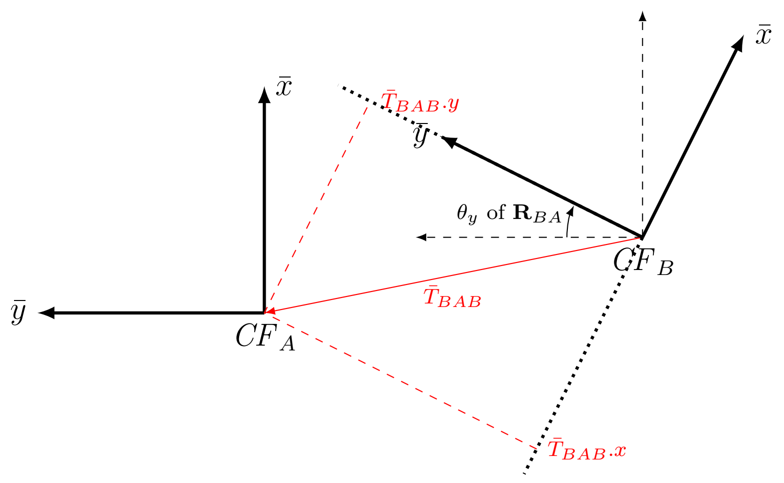


Figure 3.4: Nomenclature: coordinate frame transformations.

5. Individual elements of 3D vectors are referred to as $x$, $y$, or $z$ whereas the elements of the 2D vectors in the image plane are referred to as horizontal ($h$) and vertical ($v$) to avoid confusion. Refer to Figures 3.1 and 3.3.
6. All matrix multiplications in this work are row-major pre-multiplications.
7. All vectors are column vectors.
8. The range of $\cos^{-1}$ is taken to be $[0, \pi)$.
9. The range of $\sin^{-1}$ is taken to be $[\frac{-\pi}{2}, \frac{\pi}{2})$.
10. The range of $\tan^{-1}$ is taken to be $[\frac{-\pi}{2}, \frac{\pi}{2})$.
11. The symbol '$\bullet$' is used to denote the dot product of two vectors.
12. The symbol '$\otimes$' is used to denote the cross product of two vectors.
13. The expression '$\max(x_0, x_1, \dots)$' returns the largest of its parameters.
14. The expression '$\min(x_0, x_1, \dots)$' returns the smallest of its parameters.
15. The expression '$\underset{x}{\operatorname{argmin}}\,[f(x)]$' returns the value of $x$ which results in the minimum value of $f(x)$.
16. The expression '$\underset{x}{\operatorname{argmax}}\,[f(x)]$' returns the value of $x$ which results in the maximum value of $f(x)$.

## 3.2 Numerical optimisation

This section provides information on how the sets of parameters which yield a minimum output of a given function are determined. Section 2.2 discusses the attributes of the different algorithms discussed here and their application to photogrammetric calibration. These functions typically have many parameters and are not known explicitly, meaning that their derivatives can not be directly computed but must be estimated.

If a near-optimal initial set of parameters is known, then these parameters can be used as a starting point and intelligently perturbed to find the best set of parameters near to the initial set. If there is insufficient prior knowledge to determine an initial parameter set then a global search must be performed. These techniques are typically computationally expensive and do not yield high precision results for black box cost functions. The results of global optimisation (Sec-

Table 3.1: Genetic algorithm parameters.

| | **Scenario** | |
|---|---|---|
| | **Lens distortion** | **Pose estimation** |
| Number of parameters | 10 | 6 |
| Population size | 300 | 300 |
| Generations | 300 | 300 |
| Tourney size | 5 | 5 |
| Breeding probability | 0.8 | 0.6 |
| Mutation probability | 0.05 | 0.25 |

tion 3.2.1) are typically suitable for further refinement by a local optimisation routines (Section 3.2.2).

### 3.2.1 Coarse global optimisation

This section presents the details of the genetic algorithm [31] used for global optimisation in this work. A broader overview of global optimisation is provided in Section 2.2.2. For a genetic algorithm implementation the following parameters need to be defined:

1. **Parameter ranges.** These are the valid continuous intervals over which each parameter may vary and still constitute a feasible solution.
2. **Mapping of a solution to an individual's chromosome.** In this work only real valued problems are considered. An individual's chromosome is the list of real parameters. Each gene (the smallest unit manipulated by a genetic algorithm) is a real floating point value.
3. **Quantification of an individual's fitness.** The cost functions optimised in this work all have a minimum possible value of 0.0. The fitness is an inverse sigmoidal function such that a Cost Function Result (CFR) of zero corresponds to a fitness of 1.0 and decreases to a fitness of 0.0 for a CFR of $\infty$.
4. **Fitness biased manner of individual selection for breeding.** A tournament selection strategy was used for this work. A specified number ($N$) of individuals from the population are selected at random, thereafter

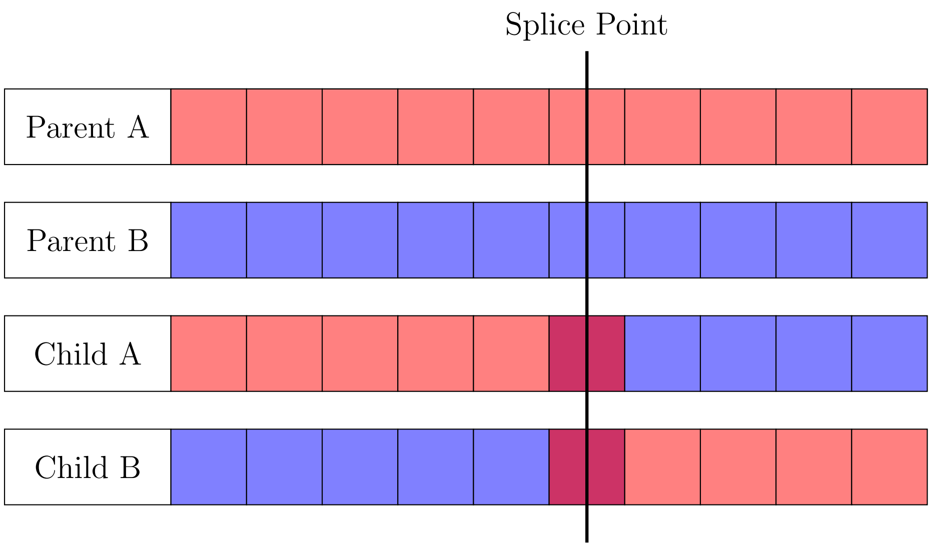


Figure 3.5: Genetic algorithm: breeding.

the two fittest in that pool are used for breeding. Larger values of $N$ favour the fittest individuals but may adversely affect the searching of outlying areas for potentially better minima. All individuals are returned to the pool for possible future selection afterwards. Table 3.1 has the tourney sizes used in this work.

5. **Probability of breeding or cloning the selected breeding pair.** Each breeding pair is randomly either cloned or bred to create two new individuals for the next generation. Table 3.1 has the breeding probabilities used in this work.
6. **Creation of new individuals via breeding.** A splice point, i.e. a gene number, was chosen at random for each breeding. The first child contains genes from the first parent up to the splice point and genes from the second parent after the splice point (vice versa for the second child). At the splice point the average of the genes for the two parents is taken as illustrated by Figure 3.5.
7. **The chance and effects of mutation when breeding.** Whenever a breeding takes place each gene has a chance that it will be mutated. Mutated genes are set to a random value in the valid range for that gene. This serves to create outliers in the population which may find better solutions.

Table 3.1 has the mutation probabilities used in this work.

8. **Optimisations and modifications.** Two modifications were used in this work. The first was a standard technique called 'elitism' which prevents regression by always copying the best individual, unaltered, to the next generation. The second optimisation is an adaptation of elitism termed 'dominance' which implements basic hill climbing. The best solution is recopied twice, once with a small positive value added to the first gene and once with this small value subtracted from the first gene. Repeating this for all $M$ genes creates a total of $2M$ slightly varying offspring of the best individual. At least one of these offspring will be an improvement if the best individual is not at an exact minimum.

With the above genetic algorithm operations and parameters defined, one randomly generates the first population using the specified valid ranges for the parameters. The fitness of each individual is then assessed and the next generation seeded with the individuals obtained from elitism and dominance. Thereafter selection and breeding/cloning/mutation are repeatedly performed until the next generation has the required number of intervals. The original generation is then deleted and the current generation used to create the next generation. This process is repeated until either a suitably fit individual is found or the maximum number of generations is exceeded. No libraries were used for the global optimisations, all routines are the author's own work.

Table 3.1 shows typical parameters used for the genetic algorithms used. It was seen that lens distortion, despite having many parameters, had a stronger minimum than pose optimisation. Thus a higher breeding probability and lower mutation chance were used for distortion characterisation resulting in more individuals to clustering in the vicinity of the minimum and the search thereof. Conversely, mutations were encouraged in the pose estimation to stimulate searching the entire problem space. This had the effect of having a superior fittest individual but at the expense of significantly decreased average population fitness.

### 3.2.2 Local numeric refinement

This section presents the mathematical workings of 5 mathematical algorithms: 3 classical and 2 more recent. All of the algorithms considered were gradient based methods. A description of the characteristics of the algorithms are provided in Section 2.2.1. No libraries were used for local optimisations, all routines used are the author's own original work. The formulations of the optimisation routines are the author's own conversions from typical algorithmic pseudo-code listings to single equations for ease of reference.

#### 3.2.2.1 One dimensional optimisation

In a 1D search for a local minimum, one needs either an interval in which to expect the minimum, or a starting point and maximum step size per iteration. In the case of a known interval, i.e. the minimum is in the interval $(a^0, b^0)$, the golden ratio (which is the limit of the ratio of successive numbers in the Fibonacci sequence) search is near optimal [69]. The idea is to successively test whether the minimum is in the left or right portion (as divided by the golden ratio) of the interval. The relevant part of the interval is retain and subdivided again according to the ratio. This iterative procedure is followed until either sufficient iterations have passed or the interval is deemed suitably small. The midpoint of the final interval is taken as the solution. Equation 3.1 provides the iterative step:

$$\begin{bmatrix} a^{k+1} \\ b^{k+1} \\ \lambda_1^{k+1} \\ \lambda_2^{k+1} \end{bmatrix} = \mathbf{A} \begin{bmatrix} a^k \\ b^k \\ \lambda_1^k \\ \lambda_2^k \end{bmatrix} \tag{3.1}$$

where

$(a^i, b^i)$ = search interval at iteration $i$,

$(a^0, b^0)$ = the initial specified search interval,

$\lambda_j^i$ = golden ratio search point $j$ at iteration $i$,

$$\lambda_1^0 = \text{initial golden ratio search point 1: } a^0 + \gamma^2(b^0 - a^0),$$
$$\lambda_2^0 = \text{initial golden ratio search point 2: } a^0 + \gamma(b^0 - a^0),$$
$$k = \text{the current iteration number,}$$
$$\gamma = \text{the golden ratio:} \frac{\sqrt{5}-1}{2}, \text{ and}$$
$$\mathbf{A} = \begin{cases} \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \gamma & 1-\gamma & 0 \end{bmatrix} & \text{if } f(\lambda_1^k) > f(\lambda_2^k), \\ \\ \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 1-\gamma^2 & 0 & 0 & \gamma^2 \\ 0 & 0 & 1 & 0 \end{bmatrix} & \text{if } f(\lambda_1^k) \leq f(\lambda_2^k). \end{cases}$$

If an interval is not known, then a heuristic method to find an interval bracketing the minimum can be used. Powell's quadratic interpolation, described below, is an example of such an algorithm. This algorithm uses the three most recent estimates to construct a second order polynomial, whose turning point is then determined. If the turning point is too far from the current point to be trusted with certainty (i.e. it is more than the specified maximum step size away) then the next estimate is merely a step of the maximum size in the descent direction. Equation 3.2 provides the initialisation of the algorithm by selecting three initial points based on the maximum step size and initial point:

$$\lambda^1 = \lambda^0 + h \tag{3.2}$$
$$\lambda^2 = \begin{cases} \lambda^0 - h & \text{if } f(\lambda^1) \leq f(\lambda^2) \\ \lambda^0 + 2h & \text{if } f(\lambda^1) > f(\lambda^2) \end{cases}$$

where

$$(\lambda_1^1, \lambda_1^2) = \text{the first search interval,}$$
$$\lambda_0 = \text{the original starting position, and}$$
$$h = \text{the specified step size.}$$

Using the three latest points, the first and second derivatives are derived and used to solve the turning point of the quadratic polynomial passing through the points. This turning point is then a candidate point to be used in subsequent iterations of Powell's algorithm. Equation 3.3 shows this calculation of the next estimate from the three latest estimates:

$$\lambda^{k+1} = \frac{f''^{\dagger}(\lambda^{k-1} - \lambda^{k}) - f'^{\dagger}}{2f''^{\dagger}} \tag{3.3}$$

where

$$f''^{\dagger} = \frac{2}{\lambda^{k-2} + \lambda^{k-1}} \left( \frac{f(\lambda^{k-2}) - f(\lambda^{k-1})}{\lambda^{k-2} - \lambda^{k-1}} - \frac{f(\lambda^{k-1}) - f(\lambda^{k})}{\lambda^{k-1} - \lambda^{k}} \right)$$

$$f'^{\dagger} = \frac{f(\lambda^{k-1}) - f(\lambda^{k})}{\lambda^{k-1} - \lambda^{k}}.$$

Figure 3.6 provides the high level flow of the Powell's algorithm using the initialisation and update steps provided by Equations 3.2 and 3.3 respectively. The algorithm checks for and rectifies candidate points provided by Equation 3.3 that are beyond the stipulated maximum distance from the current estimation of the minimum. With regard to Figure 3.6 '$H$' is the maximum step size and '$\lambda^{i}$' is the $i^{\text{th}}$ estimation of the minimum.

#### 3.2.2.2 Steepest descent

Steepest descent is the oldest local optimisation routine. At each iteration the direction of steepest descent is determined. Thereafter that direction is traversed until a minimum is found. Then a new descent direction is determined and the process repeated until either a suitable minimum is found or a maximum number of iterations exceeded.

The direction of steepest descent is merely the negative of the vector of first

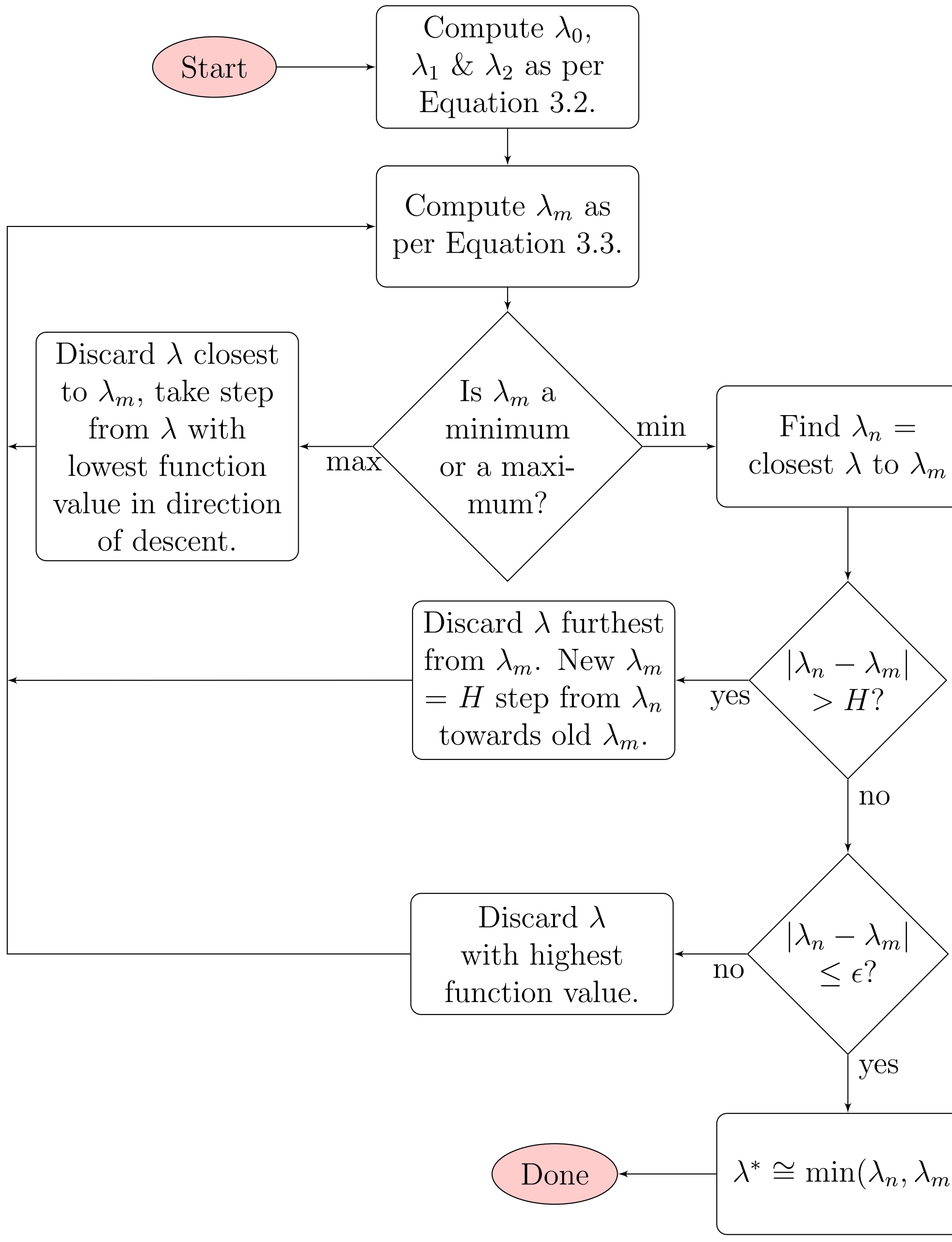

Start
Compute $\lambda_0$, $\lambda_1$ & $\lambda_2$ as per Equation 3.2.
Compute $\lambda_m$ as per Equation 3.3.
Is $\lambda_m$ a minimum or a maximum?
max
min
Discard $\lambda$ closest to $\lambda_m$, take step from $\lambda$ with lowest function value in direction of descent.
Find $\lambda_n$ = closest $\lambda$ to $\lambda_m$
$|\lambda_n - \lambda_m| > H$?
yes
Discard $\lambda$ furthest from $\lambda_m$. New $\lambda_m$ = $H$ step from $\lambda_n$ towards old $\lambda_m$.
no
$|\lambda_n - \lambda_m| \leq \epsilon$?
no
Discard $\lambda$ with highest function value.
yes
$\lambda^* \cong \min(\lambda_n, \lambda_m)$
Done

Figure 3.6: Flow diagram of Powell's interpolation algorithm.

order partial derivatives:

$$\bigtriangledown \mathrm{C}(\bar{x}) = \begin{bmatrix} \frac{\delta}{\delta \bar{x}_1}\mathrm{C}(\bar{x}) \\ \frac{\delta}{\delta \bar{x}_2}\mathrm{C}(\bar{x}) \\ \vdots \\ \frac{\delta}{\delta \bar{x}_N}\mathrm{C}(\bar{x}) \end{bmatrix} \tag{3.4}$$

where

$$\begin{aligned} \bar{x} &= \text{the point at which the gradient vector is sought,} \\ \mathrm{C}(\bar{x}) &= \text{the cost function being minimised,} \\ \bigtriangledown \mathrm{C}(\bar{x}) &= \text{the direction of steepest ascent,} \\ \frac{\sigma}{\sigma \bar{x}_i}\mathrm{C}(\bar{x}) &= \text{the partial derivative of C w.r.t. parameter } i \text{ at } \bar{x}\text{, and} \\ N &= \text{the number of parameters in the set being optimised.} \end{aligned}$$

Once a search direction has been chosen, the multidimensional problem has been reduced to a linear one and the minimum can be found using the methods described in Section 3.2.2.1. Typically the maximum step size that may be taken is limited, due to uncertainty of the gradient vector. The entire steepest procedure can be expressed mathematically as:

$$\bar{x}^{k+1} = \bar{x}^k + \alpha^{k'} \bar{U}^k \tag{3.5}$$

where

$$\begin{aligned} \bar{x}^{k+1} &= \text{next refinement of the parameters,} \\ k &= \text{the current iteration number,} \\ \bar{x}^k &= \text{current parameters,} \\ \bar{U}^k &= \text{unit direction in which to search,} \\ &= \frac{- \bigtriangledown \mathrm{C}(\bar{x}^k)}{\| \bigtriangledown \mathrm{C}(\bar{x}^k)\|} \\ \bigtriangledown \mathrm{C}(\bar{x}^k) &= \text{the gradient vector as per Equation 3.4,} \\ \mathrm{C}(\bar{x}) &= \text{the cost function being minimised,} \\ \alpha^{k'} &= \text{the clipped distance to go in the search direction,} \end{aligned}$$

$$= \begin{cases} \alpha^k & \text{if } \alpha^k < \alpha^{\max} \\ \alpha^{\max} & \text{if } \alpha^k \geqslant \alpha^{\max} \end{cases},$$

$$\alpha^k = \underset{\alpha}{\operatorname{argmin}} \left[ \mathrm{C}\left( \bar{x}^k + \alpha \bar{U}^k \right) \right] \text{ as per Section 3.2.2.1, and}$$

$$\alpha^{\max} = \text{maximum step size to be taken per iteration.}$$

### 3.2.2.3 Fletcher-Reeves conjugate gradient algorithm

Conjugate gradient methods aim to improve the convergence speed of steepest descent without resorting to second order derivative information. They do this by trying to improve new search directions, based on the previous search directions. This is because steepest descent search directions tend to slowly meander down valleys to the minimum as explained in Section 2.2.1 and depicted in Figure 2.1. The basic procedure is identical to that of steepest descent (Equation 3.5), except that the search direction is not merely the negative of the normalised gradient vector. For Fletcher-Reeves conjugate gradient determination, a multiple of the previous search direction is added to the current search direction. This multiple is set to either the ratio of the improvement in gradient magnitudes, or periodically set to zero to avoid over compensation. Equation 3.6 shows how to calculate the Fletcher-Reeves [21] search directions:

$$\bar{U}^k = \frac{\bar{V}^k}{\|\bar{V}^k\|} \tag{3.6}$$

where

$$\bar{U}^k = \text{the unit direction in which to search,}$$

$$\bar{V}^k = \frac{- \triangledown \mathrm{C}(\bar{x}^k)}{\| \triangledown \mathrm{C}(\bar{x}^k)\|} + \beta^k \bar{V}^{k-1},$$

$$\triangledown \mathrm{C}(\bar{x}^k) = \text{the gradient vector as per Equation 3.4,}$$

$$\beta^k = \text{the coefficient of previous search directions}$$

$$= \begin{cases} \frac{\|\triangledown \mathrm{C}(\bar{x}^k)\|^2}{\|\triangledown \mathrm{C}(\bar{x}^{k-1})\|^2} & \text{if } k \bmod (N+1) \neq 0 \\ 0 & \text{if } k \bmod (N+1) = 0 \end{cases}, \text{ and}$$

$$N = \text{the number of parameters in the set being optimised.}$$

#### 3.2.2.4 Leapfrog algorithm

The Leap Frog Algorithm (LFA) uses the gradient information (i.e. Equation 3.4) to determine the acceleration induced by a particle for a specified time step. The particle's velocity and position are monitored. When the velocity decreases (because the particle is going, i.e. away from a local minimum) its velocity is decreased rather than zeroed. This is what gives LFA its ability to avoid shallow local minima and find stronger minima. Heuristics are used to control the time step from iteration to iteration to provide better resolution in determining the minimum. The complete algorithm, redrawn from Snyman [19], is given in flow diagram form in Figure 3.7.

## 3.3 Rotation formalisms

Rotation matrices are the default method of performing rotations used in this research, due to their ease of use and manipulation with linear algebraic techniques. The following subsections describe the properties of these matrices and how they relate to other common methods used to express rotations. More information can be found in Hartley and Zisserman [70].

### 3.3.1 Rotation matrix

A rotation matrix is a $3 \times 3$ matrix that, when used to multiply a 3 element vector, changes the CF in which the vector is expressed. That is, the magnitude of the vector is unchanged but its elements are changed. Stated in terms of the notation defined in Section 3.1.2, a rotation matrix changes the third subscript of a vector but does not change the two points whose spatial difference the vector embodies. This is displayed graphically for $\bar{T}_{P_1P_2A}$ and $\bar{T}_{P_1P_2B}$ in Figure 3.3 where the length of the vector is unchanged but it has different projections on the two CFs.

Any given rotation matrix, $\mathbf{R}_{jk}$, is orthogonal. From this orthogonality some

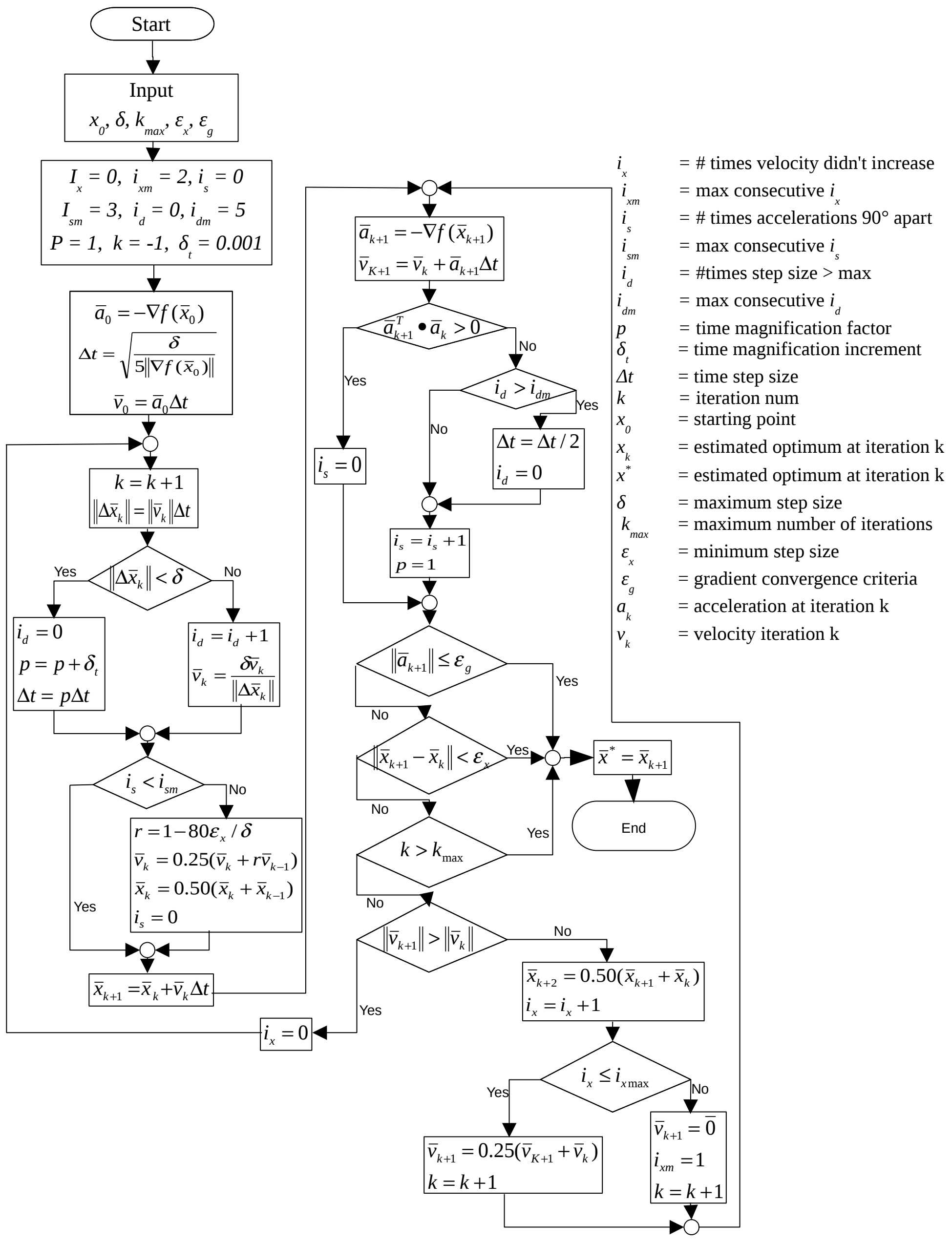

Start
Input
$x_0, \delta, k_{max}, \varepsilon_x, \varepsilon_g$
$I_x = 0,\ i_{xm} = 2,\ i_s = 0$
$I_{sm} = 3,\ i_d = 0,\ i_{dm} = 5$
$P = 1,\ k = -1,\ \delta_t = 0.001$
$\bar{a}_0 = -\nabla f(\bar{x}_0)$
$\Delta t = \sqrt{\frac{\delta}{5\|\nabla f(\bar{x}_0)\|}}$
$\bar{v}_0 = \bar{a}_0 \Delta t$
$k = k + 1$
$\|\Delta \bar{x}_k\| = \|\bar{v}_k\| \Delta t$
$\|\Delta \bar{x}_k\| < \delta$
Yes
No
$i_d = 0$
$p = p + \delta_t$
$\Delta t = p \Delta t$
$i_d = i_d + 1$
$\bar{v}_k = \frac{\delta \bar{v}_k}{\|\Delta \bar{x}_k\|}$
$i_s < i_{sm}$
No
$r = 1 - 80\varepsilon_x / \delta$
$\bar{v}_k = 0.25(\bar{v}_k + r\bar{v}_{k-1})$
$\bar{x}_k = 0.50(\bar{x}_k + \bar{x}_{k-1})$
$i_s = 0$
Yes
$\bar{x}_{k+1} = \bar{x}_k + \bar{v}_k \Delta t$
$i_x = 0$
$\bar{a}_{k+1} = -\nabla f(\bar{x}_{k+1})$
$\bar{v}_{K+1} = \bar{v}_k + \bar{a}_{k+1} \Delta t$
$\bar{a}^T_{k+1} \bullet \bar{a}_k > 0$
No
Yes
$i_d > i_{dm}$
Yes
No
$i_s = 0$
$\Delta t = \Delta t / 2$
$i_d = 0$
$i_s = i_s + 1$
$p = 1$
$\|\bar{a}_{k+1}\| \le \varepsilon_g$
Yes
No
$\|\bar{x}_{k+1} - \bar{x}_k\| < \varepsilon_x$
Yes
$\bar{x}^* = \bar{x}_{k+1}$
End
No
$k > k_{max}$
Yes
No
$\|\bar{v}_{k+1}\| > \|\bar{v}_k\|$
No
Yes
$\bar{x}_{k+2} = 0.50(\bar{x}_{k+1} + \bar{x}_k)$
$i_x = i_x + 1$
$i_x \le i_{x\max}$
Yes
No
$\bar{v}_{k+1} = 0.25(\bar{v}_{K+1} + \bar{v}_k)$
$k = k + 1$
$\bar{v}_{k+1} = \bar{0}$
$i_{xm} = 1$
$k = k + 1$
$i_x$ = # times velocity didn't increase
$i_{xm}$ = max consecutive $i_x$
$i_s$ = # times accelerations 90° apart
$i_{sm}$ = max consecutive $i_s$
$i_d$ = #times step size > max
$i_{dm}$ = max consecutive $i_d$
$p$ = time magnification factor
$\delta_t$ = time magnification increment
$\Delta t$ = time step size
$k$ = iteration num
$x_0$ = starting point
$x_k$ = estimated optimum at iteration k
$x^*$ = estimated optimum at iteration k
$\delta$ = maximum step size
$k_{max}$ = maximum number of iterations
$\varepsilon_x$ = minimum step size
$\varepsilon_g$ = gradient convergence criteria
$a_k$ = acceleration at iteration k
$v_k$ = velocity iteration k

Figure 3.7: Leapfrog algorithm flow diagram.

important properties about the rotation matrix may be derived:

1. $\mathbf{R}_{kj}$ is the inverse matrix of $\mathbf{R}_{jk}$ and the inverse of a rotation matrix is equal to its transpose:
$$\mathbf{R}_{kj} = \mathbf{R}_{jk}^{-1} = \mathbf{R}_{jk}^{T}. \tag{3.7}$$
2. The magnitude of each row and of each column in $\mathbf{R}_{jk}$ is exactly unity.
3. Each row of $\mathbf{R}_{jk}$ is orthogonal to every other row, i.e. their dot products are exactly zero.
4. Each column of $\mathbf{R}_{jk}$ is orthogonal to every other column, i.e. their dot products are exactly zero.

### 3.3.2 Euler angles

If the Euler yaw, pitch, and roll angles of one CF relative to another are known, then a rotation matrix can be created from these angles (i.e. $\mathfrak{f}^{\mathrm{EtoR}}$) as per Equation 3.8:

$$\begin{aligned}\mathbf{R}_{jk} &= \mathfrak{f}^{\mathrm{EtoR}}(\theta_y, \theta_p, \theta_r) \\ &= \begin{bmatrix} C_{\theta_y}C_{\theta_p} & -C_{\theta_y}S_{\theta_p}S_{\theta_r} - C_{\theta_r}S_{\theta_y} & -C_{\theta_y}S_{\theta_p}C_{\theta_r} + S_{\theta_y}S_{\theta_r} \\ S_{\theta_y}C_{\theta_p} & C_{\theta_y}C_{\theta_r} - S_{\theta_y}S_{\theta_p}S_{\theta_r} & -C_{\theta_y}S_{\theta_r} - S_{\theta_y}S_{\theta_p}C_{\theta_r} \\ S_{\theta_p} & C_{\theta_p}S_{\theta_r} & C_{\theta_p}C_{\theta_r} \end{bmatrix}\end{aligned} \tag{3.8}$$

where

$\mathbf{R}_{jk} =$ the rotation matrix converting vectors from CF $j$ to CF $k$,

$\theta_y =$ the yaw of CF $j$ relative to CF $k$,

$\theta_p =$ the pitch of CF $j$ relative to CF $k$,

$\theta_r =$ the roll of CF $j$ relative to CF $k$,

$C_{\theta_x} = \cos(\theta_x)$ where: $x \in [y, p, r]$, and

$S_{\theta_x} = \sin(\theta_x)$ where: $x \in [y, p, r]$.

There are many different possibilities for Euler angles, as the final matrix given above is the product of three rotations, each about a single axis. Any permutation of single axis rotations is viable so long as the same axis is not rotated about consecutively. Thus one needs to know which axis each angle corresponds to and in what order the single axis rotation matrices must be applied. This work uses the most common approach. First rotation is performed around the $Z$ axis, then around the $Y$ axis and then around the $X$ axis using the yaw, pitch and roll angles respectively:

$$\mathbf{R}_{ab} = \mathbf{R}_Z(\theta_y)\mathbf{R}_Y(\theta_p)\mathbf{R}_X(\theta_r) \tag{3.9}$$

where

$$\mathbf{R}_X(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix},$$

$$\mathbf{R}_Y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}, \text{ and}$$

$$\mathbf{R}_Z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

The Euler rotation angles can be obtained from a $3 \times 3$ rotation matrix as follows:

$$\begin{bmatrix} \theta_y \\ \theta_p \\ \theta_r \end{bmatrix} = \mathfrak{f}^{\mathrm{RtoE}}\left(\mathbf{R}_{jk}\right) = \begin{bmatrix} \tan^{-1}\left(\frac{\mathbf{R}_{jk}(2,1)}{\mathbf{R}_{jk}(1,1)}\right) \\ \sin^{-1}\left(\mathbf{R}_{jk}(3,1)\right) \\ \tan^{-1}\left(\frac{\mathbf{R}_{jk}(3,2)}{\mathbf{R}_{jk}(3,3)}\right) \end{bmatrix}. \tag{3.10}$$

It is not possible to directly add or subtract Euler angles or to directly use Euler angles to directly rotate a vector. Euler angles may be used in calculations after they have been converted in to a rotation matrix using Equation 3.8.

### 3.3.3 Angle axis

Euler's rotation theorem [71] tells us that any number of rotations of a rigid body can be expressed as a single rotation around some axis. The magnitude of this rotation and its corresponding vector can be calculated from a rotation matrix using Equation 3.11:

$$
\begin{aligned}
(\theta, \bar{u}) &= \mathfrak{f}^{\text{RtoAA}}\left(\mathbf{R}_{jk}\right) \qquad (3.11)\\
&\text{where}\\
\theta &= \text{the angle of rotation}\\
&= \cos^{-1}\left(\frac{\mathbf{R}_{jk}(1,1)+\mathbf{R}_{jk}(2,2)+\mathbf{R}_{jk}(3,3)-1}{2}\right), \text{ and}\\
\bar{u} &= \text{the UV around which to perform the rotation}\\
&= \begin{bmatrix} \frac{\mathbf{R}_{jk}(3,2)-\mathbf{R}_{jk}(2,3)}{2\sin(\theta)} \\ \frac{\mathbf{R}_{jk}(1,3)-\mathbf{R}_{jk}(3,1)}{2\sin(\theta)} \\ \frac{\mathbf{R}_{jk}(2,1)-\mathbf{R}_{jk}(1,2)}{2\sin(\theta)} \end{bmatrix}.
\end{aligned}
$$

This concept is similar to quaternions. A rotation matrix can be created from the angle-axis representation using Equation 3.12:

$$
\begin{aligned}
\mathbf{R}_{jk} &= \mathfrak{f}^{\text{AAtoR}}\left(\theta, \bar{U}\right) \qquad (3.12)\\
&= \begin{bmatrix}
C_\theta + \bar{U}_x^2\left(1-C_\theta\right) & \bar{U}_x\bar{U}_y\left(1-C_\theta\right) - \bar{U}_zS_\theta & \bar{U}_x\bar{U}_z\left(1-C_\theta\right) + \bar{U}_yS_\theta \\
\bar{U}_x\bar{U}_y\left(1-C_\theta\right) + \bar{U}_zS_\theta & C_\theta + \bar{U}_y^2\left(1-C_\theta\right) & \bar{U}_y\bar{U}_z\left(1-C_\theta\right) - \bar{U}_xS_\theta \\
\bar{U}_x\bar{U}_z\left(1-C_\theta\right) - \bar{U}_yS_\theta & \bar{U}_y\bar{U}_z\left(1-C_\theta\right) + \bar{U}_xS_\theta & C_\theta + \bar{U}_z^2\left(1-C_\theta\right)
\end{bmatrix}\\
&\text{where}\\
\theta &= \text{the angle of rotation,}\\
C_\theta &= \cos\left(\theta\right),\\
S_\theta &= \sin\left(\theta\right), \text{ and}\\
\bar{U} &= \text{the UV that is the rotation axis.}
\end{aligned}
$$

### 3.3.4 ABB IRB120 robot angles

The ABB IRB120 robot arm [72] used in this work (Section 5.1) provides three angles to quantify the orientation of the end effector relative to its base. Clearly the correct interpretation of these angles is paramount for all the calibration routines (Chapter 5), as they use the robot arm as a physical reference. However the angles are referred to in the IRB120 user manual only as Rx, Ry and Rz. No indication of which direction is positive and whether 0° is straight up or straight down, or of the correct order of multiplication of the matrices was provided. Preliminary observation confirmed the angles were provided in units of degrees.

The correct interpretation of these angles is necessary to assess whether the robot arm has moved sufficiently close to the desired pose. Therefore, an experiment to yield the correct interpretation of these angles was devised. The robot was commanded to a sequence of poses well within the ranges of the joints and it was ensured that no gimbal lock or other movement errors were reported. The corresponding requested orientations (the position part of the pose was ignored) and returned orientation in Rx, Ry and Rz format was recorded.

In total 10368 possible interpretations of the three values were tested. These included all permutations of 3 angles, 3 axes of rotation, 2 possible positive rotation directions, using either $\theta$ or $90° - \theta$, 6 possible ways to order the rotation matrices, and each matrix possibly being transposed. The determined correct manner of interpreting the robot arm's angles is given by Equation 3.13:

$$\mathbf{R}_{ar} = \mathfrak{f}^{\mathrm{IRBtoR}}\left(\mathrm{Rx}, \mathrm{Ry}, \mathrm{Rz}\right) = \mathbf{R}_Z(\mathrm{Rz})\mathbf{R}_Y(90° - \mathrm{Ry})\mathbf{R}_Z(-\mathrm{Rx}) \tag{3.13}$$

where

$\mathbf{R}_{ar}$ = rotation matrix converting from the arm CF to the robot CF,

$\mathbf{R}_Y, \mathbf{R}_Z$ = single axis rotation matrices around the $Y$ and $Z$ axes respectively as per Equation 3.9, and

Rx, Ry, Rz = the reported orientation of the arm.

### 3.3.5 Creating a rotation matrix from a set of orthogonal vectors

If $\bar{U}_{jxk}$ is a UV known to point along the $X$ axis of CF $j$ but is expressed in terms of its projection on CF $k$, and equivalent vectors $\bar{U}_{jyk}$ and $\bar{U}_{jzk}$ are known for the $Y$ and $Z$ axes, then $\mathbf{R}_{jk}$ can be calculated from the three vectors. This rotation definition is used in the PMJ (Section 5.4.1), LED offset (Section 5.4.2), and focal length (Section 5.5.3) calibrations as well as the helmet tracker application (Section 6.2). Equation 3.14 shows how to create the rotation matrix from the vectors:

$$\mathbf{R}_{jk} = \mathfrak{f}^{\text{UVtoR}}\left(\bar{U}_{jxk}, \bar{U}_{jyk}, \bar{U}_{jzk}\right) = \begin{bmatrix} \bar{U}_{jxk}.x & \bar{U}_{jxk}.y & \bar{U}_{jxk}.z \\ \bar{U}_{jyk}.x & \bar{U}_{jyk}.y & \bar{U}_{jyk}.z \\ \bar{U}_{jzk}.x & \bar{U}_{jzk}.y & \bar{U}_{jzk}.z \end{bmatrix} \tag{3.14}$$

where

$\mathbf{R}_{jk}$ = rotation matrix to change vectors from being expressed in CF $j$ to being expressed in CF $k$,

$\bar{U}_{jxk}$ = projection of CF $j$'s X axis on to CF $k$,
$= \left[\bar{U}_{jxk}.x, \bar{U}_{jxk}.y, \bar{U}_{jxk}.z\right]^T$

$\bar{U}_{jyk}$ = projection of CF $j$'s Y axis on to CF $k$,
$= \left[\bar{U}_{jyk}.x, \bar{U}_{jyk}.y, \bar{U}_{jyk}.z\right]^T$

$\bar{U}_{jzk}$ = projection of CF $j$'s Z axis on to CF $k$, and
$= \left[\bar{U}_{jzk}.x, \bar{U}_{jzk}.y, \bar{U}_{jzk}.z\right]^T$.

## 3.4 Fundamental operations

This section details several mathematical building blocks which are used extensively in subsequent chapters. All non-trivial mathematics required in subsequent chapters are presented in the form of a single equation which can easily be referenced.

### 3.4.1 The difference between two rotation matrices

The difference matrix between two rotation matrices which share a common reference frame can be calculated by multiplying one of the rotations by the inverse of the other:

$$\mathbf{R}_{jk} = \mathbf{R}_{kl}^T \mathbf{R}_{jl} \tag{3.15}$$

where

$\mathbf{R}_{jk}$ = desired matrix to convert vectors from CF $j$ to CF $k$,

$\mathbf{R}_{kl}$ = matrix to convert vectors from CF $k$ to CF $l$, and

$\mathbf{R}_{jl}$ = matrix to convert vectors from CF $j$ to CF $l$.

It is often required to quantify the similarity of two rotation matrices. An example of this is the focal length calibration (Section 5.5.3) which needs to determine the similarity of many poses to find out when they are all maximally alike. Determining the similarity between rotations involves first finding the rotation difference matrix between the two rotation matrices and then either finding the smallest possible rotation angle using Equation 3.11 or calculating the Euler angles using Equation 3.10. Equation 3.16 portrays the former option:

$$\theta = \mathfrak{f}^{\text{RtoAA}} \left( {}^{m}\mathbf{R}_{jk}^{T} \, {}^{t}\mathbf{R}_{jk} \right) \tag{3.16}$$

where

$\theta$ = angular difference between the rotation matrices,

$\mathfrak{f}^{\text{RtoAA}}$ = as per Equation 3.11,

${}^{m}\mathbf{R}_{jk}$ = measured rotation matrix of CF $j$ w.r.t. CF $k$, and

${}^{t}\mathbf{R}_{jk}$ = theoretical rotation matrix of CF $j$ w.r.t. CF $k$.

Note that to calculate the angular difference between the matrices, the order of multiplication and which of the two matrices is transposed is unimportant. This is not true if one calculates the Euler angles of the difference matrix.

### 3.4.2 Creating a vector from an undistorted image coordinate

A vector pointing from the camera to an object in the camera's FOV can be determined if the intrinsic camera parameters (focal length, pixel dimensions, principal point and skewness (assumed to be zero for modern imagers) of the image axes) are known. This 2D to 3D conversion is required in the Cartesian versus polar modelling (Section 4.1) and the calibration target (Section 4.2) assessments. It is also used in calibrations of the LED offset (Section 5.4.2), the focal length (Section 5.5.3), the camera w.r.t the mount pose (Section 5.5.5), and the mount pose (Section 5.5.4). Finally the applications make use of this in the stitching accuracy assessment (Section 6.1.3) and helmet tracking (Section 6.2).

Here it is assumed that lens distortion is either negligible or has already been taken into account. The image coordinates are converted to 2D spatial values relative to the principle point by using the pixel dimensions. It is important to take into account the different positive direction conventions of the image and and camera CF when doing the scaling (see Figure 3.1). In general it is not possible to determine the distance of the object from its position in a single camera's FOV and so the vector is scaled to a UV. The third dimension is the focal length of the camera. Equation 3.17 shows how this is done.

$$\begin{aligned}\bar{U}_{coc} &= \mathfrak{f}^{\text{img}\rightarrow\text{vec}}\left(\bar{I}^u_o, \bar{PP}, FLen, pix\_w, pix\_h\right) \\ &= \bar{V}_{coc}/ \parallel \bar{V}_{coc} \parallel\end{aligned} \tag{3.17}$$

where

$\bar{V}_{coc}$ = a vector ending in the inverted image plane pointing from the camera to the object,

$$= \begin{bmatrix} FLen \\ (\bar{PP}_h - \bar{I}^u_{o_h})pix\_w \\ (\bar{PP}_v - \bar{I}^u_{o_v})pix\_h \end{bmatrix},$$

$\bar{U}_{coc}$ = a UV pointing from the camera centre to the object,

$\bar{I}^u_o = [\bar{I}^u_{o_h}, \bar{I}^u_{o_v}]^T$ = the undistorted 2D image pixel position of the object,

$\bar{PP} = [\bar{PP}_h, \bar{PP}_v]^T =$ the optical axis intersection pixel position,

$pix\_w =$ the width of the pixels on the camera's imager,

$pix\_h =$ the height of the pixels on the camera's imager, and

$FLen =$ the equivalent pinhole model focal length of the camera's lens.

Note that it is required that the focal length and pixel dimensions be specified in the same units.

### 3.4.3 Determining the undistorted image coordinate of a point relative to a camera

If a point is known in the camera's CF, then it is possible to determine where the point is in the camera's FOV in the absence of lens distortion if the intrinsic parameters of the camera are also known. This 3D to 2D conversion is required in the photogrammetric stitching application (Section 6.1.1), the creation of the synthetic data (Section 7.1), and accuracy assessment (Section 7.2) of the APCCS sensitivity analysis.

The vector to the point is clipped to coincide with the image plane. Thereafter, the $Y$ and $Z$ components are used to calculate the $h$ and $v$ pixel coordinates respectively. This pixel conversion takes into account the top-left origin and image positive $h$ and $v$ positive directions by scaling the clipped vector's $y$ and $z$ elements by the negative pixel dimensions and adding the pixel coordinates of the principal point. Equation 3.18 shows this mathematically:

$$\begin{aligned}\bar{I}^u &= \mathfrak{f}^{\mathrm{P}\to\mathrm{U}}(\bar{T}_{cpc}, \bar{PP}, FLen, pix\_w, pix\_h) \\ &\equiv \mathfrak{f}^{\mathrm{P}\to\mathrm{U}}(\bar{T}_{cpc}, \bar{I}_{\mathrm{params}}) \\ &= \begin{bmatrix} \bar{I}^u_h \\ \bar{I}^u_v \end{bmatrix} = \begin{bmatrix} \bar{PP}_h - \frac{FLen}{pix\_w} \times \frac{\bar{T}_{cpc}.y}{\bar{T}_{cpc}.x} \\ \bar{PP}_v - \frac{FLen}{pix\_h} \times \frac{\bar{T}_{cpc}.z}{\bar{T}_{cpc}.x} \end{bmatrix}\end{aligned} \tag{3.18}$$

where

$\bar{I}^u =$ the desired undistorted image coordinate,

$$
\begin{aligned}
\bar{T}_{cpc} &= \text{translation of point w.r.t. camera in camera CF} \\
\bar{PP} &= \text{the optical axis intersection pixel position,} \\
pix\_w &= \text{the width of the pixels on the camera's imager,} \\
pix\_h &= \text{the height of the pixels on the camera's imager, and} \\
FLen &= \text{the exact focal length of the camera's lens.}
\end{aligned}
$$

### 3.4.4 Determining the closest point of intersection of two 3D lines

Two 3D lines in free space are unlikely to cross perfectly; instead they will have a closest point of intersection. Determining this point is required for to calibrate the LED offset (Section 5.4.2) and the camera offset w.r.t. its mounting interface (Section 5.5.5).

At the points along each line where they are the closest, the line segment between the two lines will be perpendicular to both lines. This is expressed in Equation 3.19:

$$
\begin{aligned}
0 &= \left(\bar{T}_{rp_1r} + c_1\bar{U}_{p_1V_1r} - (\bar{T}_{rp_2r} + c_2\bar{U}_{p_2V_2r})\right) \bullet \bar{U}_{p_1V_1r}, \text{ and} \\
0 &= \left(\bar{T}_{rp_1r} + c_1\bar{U}_{p_1V_1r} - (\bar{T}_{rp_2r} + c_2\bar{U}_{p_2V_2r})\right) \bullet \bar{U}_{p_2V_2r} \qquad (3.19)
\end{aligned}
$$

where

$$
\begin{aligned}
\bar{T}_{rp_1r} &= \text{a point on line 1,} \\
\bar{U}_{p_1V_1r} &= \text{unit direction vector of line 1,} \\
c_1 &= \text{distance along line 1 from } \bar{T}_{rp_1r} \text{ in direction of } \bar{U}_{p_1V_1r} \\
&\quad\ \text{where line 1 is closest to line 2,} \\
\bar{T}_{rp_2r} &= \text{a point on line 2,} \\
\bar{U}_{p_1V_1r} &= \text{unit direction vector of line 2, and} \\
c_2 &= \text{distance along line 2 from } \bar{T}_{rp_2r} \text{ in direction of } \bar{U}_{p_2V_2r} \\
&\quad\ \text{where line 2 is closest to line 1.}
\end{aligned}
$$

Simultaneously solving Equation 3.19 and then averaging the two lines' closest points yields the closest point of intersection of the lines:

$$
\begin{aligned}
\bar{T}_{rp_ir} &= \mathfrak{f}^{\text{int}}\left(\bar{T}_{rp_1r}, \bar{U}_{p_1V_1r}, \bar{T}_{rp_2r}, \bar{U}_{p_2V_2r}\right)\\
&= \frac{1}{2}\left(\bar{T}_{rp_1r} + c_1\bar{U}_{p_1V_1r} + \bar{T}_{rp_2r} + c_2\bar{U}_{p_2V_2r}\right) \qquad (3.20)
\end{aligned}
$$

where

$$
\begin{aligned}
\bar{T}_{rp_ir} &= \text{the position of the closest point of intersection,}\\
c_1 &= (\bar{T}_{rp_2r} - \bar{T}_{rp_1r}) \bullet \bar{U}_{p_1V_1r} + c_2(\bar{U}_{p_1V_1r} \bullet \bar{U}_{p_2V_2r}), \text{ and}\\
c_2 &= \frac{(\bar{T}_{rp_2r} - \bar{T}_{rp_1r}) \bullet \bar{U}_{p_2V_2r} - (\bar{U}_{p_1V_1r} \bullet \bar{U}_{p_2V_2r})(\bar{T}_{rp_2r} - \bar{T}_{rp_1r}) \bullet \bar{U}_{p_1V_1r}}{(\bar{U}_{p_1V_1r} \bullet \bar{U}_{p_2V_2r})^2 - 1}.
\end{aligned}
$$

### 3.4.5 Finding the centre of an LED image

This is a critical component in the camera calibration pipeline: all the processing steps assume that the centre of the LED images are found accurately and consistently. Determining the centre of an LED is required for the Cartesian versus polar modelling and (Section 4.1) the calibration target (Section 4.2) assessments. It is also used in calibrations of the LED offset (Section 5.4.2), the DU calibration (5.5.1), the UD calibration (Section 5.5.2), the focal length (Section 5.5.3), the camera w.r.t the mount pose (Section 5.5.5), and the mount pose (Section 5.5.4). Helmet tracking (Section 6.2) also makes use of this building block.

The first step is to threshold the image to find all pixels that have a higher intensity than the background. This thresholding can either use a fixed value or use a dynamic, adaptive threshold. Standard texts on image processing such as Gonzales and Woods [73] can be consulted for further details on thresholding methods.

After the image has been thresholded, a connected component analysis is performed. This analysis groups all adjacent pixels above the threshold resulting in a list of 'blobs' each with a list of its constituent pixels. This is typically performed recursively but deterministic and parallel [74] variants also exist. Each blob is then checked for suitability before its centre is determined. The number of pixels is checked to determine if the blob is large enough. This size check eliminates small blobs caused by noise in the image and spurious stray light. Thereafter, each blob is checked for symmetry as the blob should be fairly round if it is an image of the LED. This symmetry is determined from the ratio of the lengths of the projections of the blob's pixels onto the best fit straight line and the line perpendicular to it. If this ratio is above the stipulated threshold then the blob is saved for further processing. Equation 3.21 expresses this mathematically:

$$K_{s,th} < \frac{L_P}{L_A} \tag{3.21}$$

$$\begin{aligned}
&\text{where :}\\
K_{s,th} &= \text{minimum symmetry required for acceptance,}\\
L_P &= \text{projection length perpendicular to best fit line,}\\
&= L_{P,max} - L_{P,min}\\
L_A &= \text{projection length along best fit straight line,}\\
&= L_{A,max} - L_{A,min}\\
L_{P,max} &= \underset{i\in((0,N-1)\cap\mathbb{Z})}{\text{argmax}} \left[(\bar{P}_i - [0,c]^T) \bullet \bar{U}_P\right],\\
L_{P,min} &= \underset{i\in((0,N-1)\cap\mathbb{Z})}{\text{argmin}} \left[(\bar{P}_i - [0,c]^T) \bullet \bar{U}_P\right],\\
L_{A,max} &= \underset{i\in((0,N-1)\cap\mathbb{Z})}{\text{argmax}} \left[(\bar{P}_i - [0,c]^T) \bullet \bar{U}_L\right],\\
L_{A,min} &= \underset{i\in((0,N-1)\cap\mathbb{Z})}{\text{argmin}} \left[(\bar{P}_i - [0,c]^T) \bullet \bar{U}_L\right],\\
\bar{P}_i &= \text{the coordinate } i^{th} \text{ pixel in the blob,}\\
\bar{U}_L &= \frac{\bar{V}_L}{\|\bar{V}_L\|},\\
\bar{V}_L &= [1,m]^T,\\
\bar{U}_P &= \frac{\bar{V}_P}{\|\bar{V}_P\|},\\
\bar{V}_P &= \begin{bmatrix} 1 \\ \frac{-1}{m} \end{bmatrix},
\end{aligned}$$

$$N = \text{the number of pixels in the blob, and}$$
$$m, c = \text{parameters of best fit line as per Equation 3.22.}$$

Equation 3.21 requires the best fit straight line through the blob. This is calculated by fitting the Least Square Error (LSE) line to the blob's pixels as follows:

$$\bar{x} = \left(A^T A\right)^{-1} A^T \bar{B} \tag{3.22}$$

where

$$A = \begin{bmatrix} h_0 & 1 \\ h_1 & 1 \\ \vdots & \vdots \\ h_{N-1} & 1 \end{bmatrix}, \quad \bar{B} = \begin{bmatrix} v_0 \\ v_1 \\ \vdots \\ v_{N-1} \end{bmatrix},$$

$$\begin{aligned}(h_i, v_i) &= i^{th} \text{ pixel of the component,} \\ i &\in (0, N-1) \cap \mathbb{Z}, \\ N &= \text{the number of pixels in the blob and} \\ \bar{x} &= [m, c]^T \\ &= \text{the coefficients for the line } v = mh + c.\end{aligned}$$

Once a blob has passed the acceptance criteria for size and symmetry, it is processed to find its centre. Currently two methods for doing this are used. The first method involves finding the centroid or Centre of Gravity (COG) of the blob by finding the pixel intensity weighted average pixel coordinates:

$$H_{COG} = \frac{\sum_{i=0}^{N-1} h_i I(h_i, v_i)}{\sum_{i=0}^{N-1} I(h_i, v_i)}, \tag{3.23}$$
$$V_{COG} = \frac{\sum_{i=0}^{N-1} v_i I(h_i, v_i)}{\sum_{i=0}^{N-1} I(h_i, v_i)},$$

where

$$\begin{aligned}I(h, v) &= \text{intensity of pixel located at } (h, v), \\ (h_i, v_i) &= \text{coordinates of the component's } i^{th} \text{ pixel,} \\ i &\in (0, N-1) \cap \mathbb{Z}, \text{ and}\end{aligned}$$

$$N = \text{number of pixels in the blob.}$$

The second method is to find the best fit ellipse to the centroid. This attempts to find the smallest ellipse which simultaneously has:

1. the smallest area,
2. the sum of intensities of constituent pixels (or parts thereof) which most closely matches the sum of intensities in the blob, and
3. the greatest difference between the sum of intensities within the ellipse versus the sum of intensities in the vicinity outside the ellipse.

In this work the immediate vicinity is defined as the smallest horizontally aligned square bounding box that does not have any pixels belonging to the ellipse within $k$ pixels of the edges of the box. All 2D ellipses have 5 DOF typically either expressed as two foci and the sum of chords from them, or as long and short axes with a specified centre position and the long axis at a specified angle from horizontal. This work uses the latter formulation and minimises a weighted sum of the three criteria listed above. The minimisation is performed by means of the Leapfrog algorithm [26] described in Section 3.2. The cost function to be minimised is given in Equation 3.24:

$$\mathrm{C}^E(a, b, E_h, E_v, \theta) = c_0 \pi ab + c_1(CS - ES) + c_2(WS - 2ES) \tag{3.24}$$

$$\begin{aligned}
&\text{where}\\
\mathrm{C}^E &= \text{cost function to minimise inorder to fit the ellipse,}\\
(E_h, E_v) &= \text{centre of the ellipse,}\\
a, b &= \text{major and minor axes of the ellipse respectively,}\\
\theta &= \text{angle of major axis from horizontal,}\\
c_n &= \text{the } n^{th} \text{ weighting term,}\\
CS &= \text{centroid sum of intensities} = \sum\nolimits_{i=0}^{N-1} I(h_i, v_i),\\
WS &= \text{window sum } = \sum I(h, v) \quad \forall h, v \in W,\\
I(h, v) &= \text{image intensity at 2D coordinate } (h, v),\\
ES &= \text{elliptical sum,}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{h,v \in W} \begin{cases} I(h,v) & \text{if } CR \leqslant ER, \\ \alpha I(h,v) & \text{if } ER < CR \leqslant ER + 1, \\ 0 & \text{if } CR > ER + 1, \end{cases} \\
CR &= \text{Cartesian radius,} \\
&= \| < h, v > - < E_h, E_v > \|, \\
ER &= \text{elliptical radius,} \\
&= \sqrt{\frac{a^2 b^2}{(b\cos(\theta - \theta_C))^2 + (a\sin(\theta - \theta_C))^2}}, \\
\alpha &= (1 - (CR - ER)), \\
W &= h \in (E_h - (a+k), E_h + (a+k)), \\
&\quad\; v \in (E_v - (a+k), E_v + (a+k)), \\
k &= \text{size of border around ellipse for energy calculations, and} \\
\theta_C &= \text{Cartesian angle of pixel } (h,v) \text{ from ellipse centre,} \\
&= \tan^{-1}\left(\frac{v - E_v}{h - E_h}\right).
\end{aligned}
$$

The initial parameters of Equation 3.24 that are used to seed the numerical refinement are determined in the prior centroid and symmetry checking processes:

$$
\begin{aligned}
a &= L_A \text{ as per Equation 3.21,} \\
b &= L_P \text{ as per Equation 3.21,} \\
h &= H_{COG} \text{ as per Equation 3.23,} \\
v &= H_{COG} \text{ as per Equation 3.23, and} \\
\theta &= \tan^{-1}\left(m^{-1}\right) \text{ with } m \text{ as per Equation 3.22.}
\end{aligned}
$$

### 3.4.6 Applying Brown's distortion model

De Villiers et al. [16, 17] showed that, using sufficient parameters in Brown's distortion model, the lens distortion effects in both the DU direction as well as the UD direction can be effectively modelled. Separate parameter sets are required for each direction. Determining the DU and UD parameters is the sole aim of Section 5.5.1 and 5.5.2 respectively. Once these parameters are known they are used in every optical calibration of Chapter 5, both example applications of Chapter 6 and throughout the sensitivity analysis (Chapter 7).

The basic Brown model takes the input pixel coordinate and expresses it relative to the principal point. Thereafter the distance of the input point from the principal point is used to add both radial and tangential offsets to create the output pixel position. The radial and tangential offsets are in the form of polynomials dependant on the distance of the input point from the principal point. It is the coordinates of the principal point of coefficients of the radial and tangential polynomials which constitute the distortion parameters. The mathematics is shown in Equation 3.25:

$$
\begin{aligned}
\bar{P}^*_{h,v} &= \mathfrak{f}^{\text{Brown}}(\bar{P}_{h,v}, \bar{P}^C_{h,v}, R_1 \ldots R_{N_R}, T_1 \ldots T_{N_T}), \qquad (3.25)\\
&\equiv \mathfrak{f}^{\text{Brown}}(\bar{P}_{h,v}, \bar{V}_{\text{params}})\\
&\text{where}\\
\mathfrak{f}^{\text{Brown}} &= \text{Brown's distortion model [34, 35]},\\
\bar{V}_{\text{params}} &= [\bar{P}^C_h \bar{P}^C_v, K_1 \ldots K_{N_R}, P_1 \ldots P_{N_T}]^T,\\
\bar{P}^*_h &= \bar{P}_h + (\bar{P}_h - \bar{P}^C_h)(\sum_{i=1}^{N_R} R_i r^{2i}) + \Bigg((1 + \sum_{i=3}^{N_T} T_i r^{2i-4})\times\\
&\qquad \big(T_1(r^2 + 2(\bar{P}_h - \bar{P}^C_h)^2) + 2T_2(\bar{P}_h - \bar{P}^C_h)(\bar{P}_v - \bar{P}^C_h)\big)\Bigg),\\
\bar{P}^*_v &= \bar{P}_v + (\bar{P}_v - \bar{P}^C_v)(\sum_{i=1}^{N_R} R_i r^{2i}) + \Bigg((1 + \sum_{i=3}^{N_T} T_i r^{2i-4})\times\\
&\qquad \big(2T_1(\bar{P}_h - \bar{P}^C_h)(\bar{P}_v - \bar{P}^C_v) + T_2(r^2 + 2(\bar{P}_v - \bar{P}^C_v)^2)\big)\Bigg),\\
\bar{P}^*_{h,v} &= \text{output image point},\\
\bar{P}_{h,v} &= \text{input image point},\\
\bar{P}^C_{h,v} &= \text{centre of distortion},\\
R_n &= \text{N}^{\text{th}} \text{ radial distortion coefficient},
\end{aligned}
$$

$$\begin{aligned} T_n &= \text{N}^{\text{th}} \text{ tangential distortion coefficient,} \\ N_R &= \text{number of radial parameters,} \\ N_T &= \text{number of tangential parameters, and} \\ r &= \sqrt{(h_d - h_c)^2 + (v_d - v_c)^2}. \end{aligned}$$

Note that it is not possible to use one tangential parameter, either zero, or two or more tangential parameters are required. Whether Equation 3.25 distorts or undistorts an image point is dependent on what parameters are passed to it. Throughout the rest of this work passing parameters to Equation 3.25 called $\bar{DU}_{\text{params}}$ will mean that the point will be converted from the distorted domain to the undistorted domain. Similarly, the parameter vector $\bar{UD}_{\text{params}}$ will be used to convert undistorted pixels coordinates to their corresponding coordinates in the distorted domain.

### 3.4.7 Tetrahedron pose determination

Earlier in Section 3.4.2 it was stated that in general one cannot determine the distance to a point from its position in a single camera's FOV. However if at least four non-coplanar points can be accurately found and uniquely identified, and the displacements between the points in the real world are known, then not only can the distance to each of the points be calculated but so can the pose of the camera relative to the points. This optical pose measurement is used in the focal length determination (Section 5.5.3) and the helmet tracker application (Section 6.2).

Figure 3.8 shows the geometric setup of this scenario once all the four image coordinates have been converted into UVs using Equation 3.17.

To solve the pose of the tetrahedron, one must first consider the basic cosine rule given in Equation 3.26. The cosine rule expresses the length of one side of a triangle as a function of the second and third sides and the angle opposite the

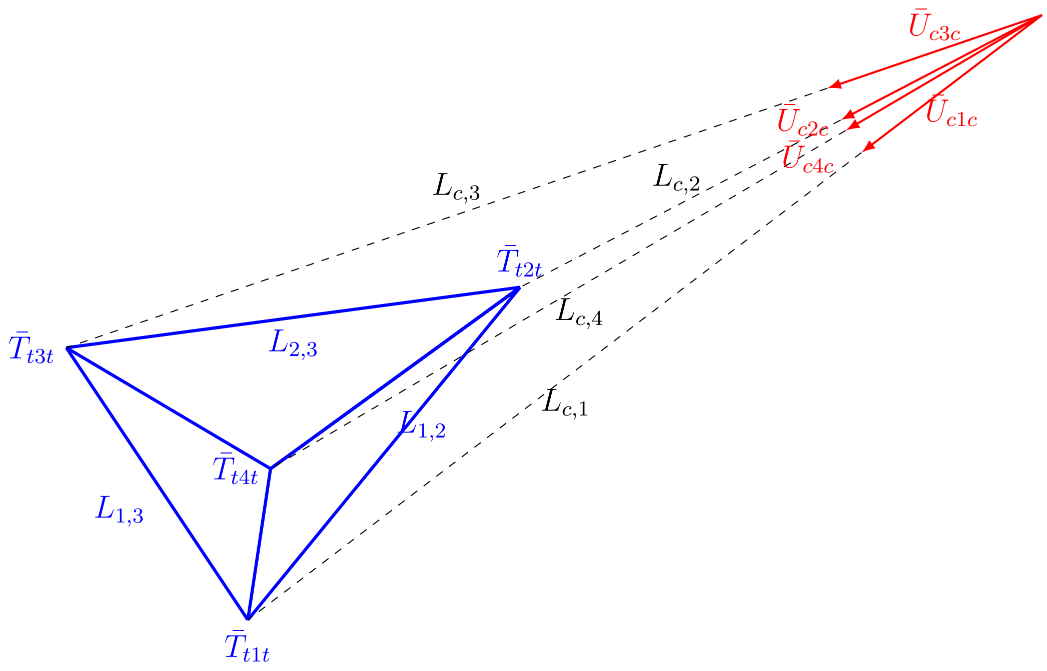


Figure 3.8: Tetrahedron pose determination.

first side:

$$a^2 = b^2 + c^2 - 2bc\cos(\theta_a) \tag{3.26}$$

where

$a, b, c =$ the lengths of the sides of a triangle, and

$\theta_a =$ the angle opposite side $a$.

Figure 3.9 illustrates the scenario applicable to Equation 3.26:

If one now considers the first two points of the tetrahedron (Figure 3.8) the distance between the two points is known, and the angle opposite the two points is determinable from the two UVs pointing to those two points. This leaves us with one equation and two unknowns. However, if we consider three points of the tetrahedron in unison we have three equations and three unknowns. Equation 3.27 expresses this situation using the identity that the cosine of the angle

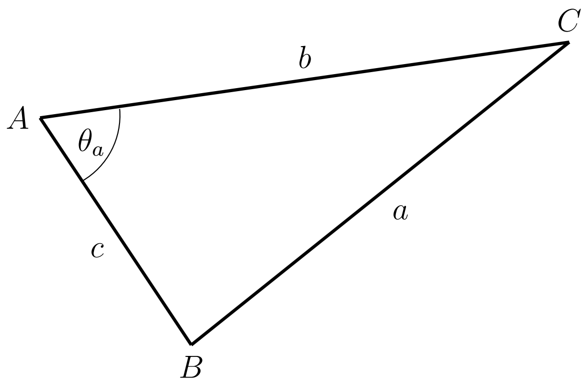


Figure 3.9: The cosine rule.

between two UVs is equal to their dot product:

$$\begin{aligned}
L_{1,2} &= L_{c,1}^2 + L_{c,2}^2 - 2L_{c,1}L_{c,2}(\bar{U}_{c1c} \bullet \bar{U}_{c2c}) \qquad (3.27)\\
L_{2,3} &= L_{c,2}^2 + L_{c,3}^2 - 2L_{c,2}L_{c,3}(\bar{U}_{c2c} \bullet \bar{U}_{c3c})\\
L_{1,3} &= L_{c,1}^2 + L_{c,3}^2 - 2L_{c,1}L_{c,3}(\bar{U}_{c1c} \bullet \bar{U}_{c3c})
\end{aligned}$$

where

$$\begin{aligned}
L_{i,j} &= \text{distance between points } i \text{ and } j \text{ of the tetrahedron } \forall i,j \in [1,3],\\
&= \|\bar{T}_{tjt} - \bar{T}_{tit}\|,\\
L_{c,i} &= \text{distance from the camera to point } i \text{ of the tetrahedron } \forall i \in [1,3],\\
&= \|\bar{T}_{cic}, \|\\
\bar{T}_{tit} &= \text{the translation of point } i \text{ in the tetrahedron CF},\\
\bar{T}_{cic} &= \text{the (unknown) translation of point } i \text{ in the camera CF},\\
\bar{U}_{cic} &= \text{the UV pointing from the camera to point } i \text{ of the tetrahedron},\\
&= \mathfrak{f}^{\text{img}\to\text{vec}}\left(\mathfrak{f}^{\text{Brown}}\left(\bar{P}_i^d, \bar{DU}_{\text{params}}\right), \bar{I}_{\text{params}}\right),\\
\bar{DU}_{\text{params}} &= \text{camera's distortion parameters},\\
\bar{I}_{\text{params}} &= \text{camera's intrinsic parameters},\\
\mathfrak{f}^{\text{img}\to\text{vec}} &= \text{as per Equation 3.17 (parameters described in Section 3.4.2)},\\
\mathfrak{f}^{\text{Brown}} &= \text{as per Equation 3.25 (parameters described in Section 3.4.6) and}\\
\bar{P}_i^d &= \text{the LED centre (Section 3.4.5) of point } i.
\end{aligned}$$

The solution to Equation 3.27 is non-trivial due to the quadratic nature of the three equations. Fischler and Bolles' [55] and Kniep et al. [56] have both presented solutions. There are always four valid solutions to Equation 3.27, although any complex solutions may be ignored. It is thus necessary to determine which solution corresponds to the truth. This is done by calculating the pose of the tetrahedron w.r.t. the camera for each solution and using that pose to calculate the translation of the fourth tetrahedron point w.r.t. the camera. This vector is then compared to the measured UV to the fourth point. The solution whose calculated fourth point most closely matches that observed by the camera, is selected. This is expressed mathematically as:

$$\begin{aligned}
\mathbf{R}_{tc}, \bar{T}_{ctc} &= \mathbf{R}_{tc,i^*}, \bar{T}_{ctc,i^*} \qquad (3.28)\\
&\text{where}\\
i^* &= \underset{i\in\mathbb{Z},[1..4]}{\operatorname{argmin}} \left[ \cos^{-1}\left( \bar{U}^*_{c4c} \bullet \left( \frac{\bar{T}_{c4c,i}}{\|\bar{T}_{c4c,i}\|} \right) \right) \right],\\
\bar{T}_{c4c,i} &= \bar{T}_{ctc,i} + \mathbf{R}_{tc,i}\bar{T}_{t4t},\\
\bar{U}^*_{c4c} &= \text{the image based vector to the 4th LED},\\
&= \mathfrak{f}^{\text{img}\rightarrow\text{vec}}\left( \mathfrak{f}^{\text{Brown}}\left( \bar{P}^d_4, \bar{DU}_{\text{params}} \right), \bar{I}_{\text{params}} \right),\\
\bar{DU}_{\text{params}} &= \text{camera's distortion parameters},\\
\bar{I}_{\text{params}} &= \text{camera's intrinsic parameters},\\
\mathfrak{f}^{\text{img}\rightarrow\text{vec}} &= \text{as per Equation 3.17 (parameters described in Section 3.4.2)},\\
\mathfrak{f}^{\text{Brown}} &= \text{as per Equation 3.25 (paremeters described in Section 3.4.6)},\\
\bar{P}^d_4 &= \text{the 4th LED's centroid (Section 3.4.5)},\\
\bar{T}_{ctc,i} &= \text{translation of tetrahedron from camera in its CF for solution } i,\\
&= \bar{T}_{c1c,i},\\
\bar{T}_{cjc,i} &= \text{translation of point } j \in [1,3] \text{ from camera in its CF for solution } i,\\
\mathbf{R}_{tc,i} &= \text{orientation of tetrahedron w.r.t. camera for solution } i,\\
&= \mathfrak{f}^{\text{UVtoR}}\left( \bar{U}_{txc,i}, \bar{U}_{tyc,i}, \bar{U}_{tzc,i} \right) \text{ (Equation 3.14)},\\
\bar{U}_{uxc,i} &= \text{the tetrahedron's } X \text{ axis expressed in camera's CF for solution } i,
\end{aligned}$$

$$
\begin{aligned}
&= \frac{\bar{T}_{c1c,i} - \frac{1}{2}\left(\bar{T}_{c2c,i} + \bar{T}_{c3c,i}\right)}{\|\bar{T}_{c1c,i} - \frac{1}{2}\left(\bar{T}_{c2c,i} + \bar{T}_{c3c,i}\right)\|}, \\
\bar{U}_{tyc,i} &= \text{the tetrahedron's } Y \text{ axis expressed in camera's CF for solution } i, \\
&= \bar{U}_{tzc,i} \otimes \bar{U}_{tyc,i}, \\
\bar{U}_{tzc,i} &= \text{the tetrahedron's } Z \text{ axis expressed in camera's CF for solution } i, \\
&= \bar{U}_{txc,i} \otimes \bar{U}^{*}_{tyc,i}, \text{ and} \\
\bar{U}^{*}_{tyc,i} &= \text{a UV in the tetrahedron's } XY \text{ plane in camera's CF for solution } i, \\
&= \frac{\bar{T}_{c3c,i} - \bar{T}_{c2c,i}}{\|\bar{T}_{c3c,i} - \bar{T}_{c2c,i}\|}.
\end{aligned}
$$

The procedure to determine the pose of a tetrahedron w.r.t. a camera involves the following steps:

1. Find the image coordinates of each point (Section 3.4.5).
2. Convert these image points to UVs (Section 3.4.2).
3. Construct and solve the simultaneous equations describing the distances of three of the tetrahedron points (Equation 3.27).
4. Use the fourth tetrahedron point to discriminate between and select the correct solution (Equation 3.28).

This is summarised in Equation 3.29:

$$
\left(\bar{T}_{tct}, \mathbf{R}_{ct}\right) = \mathfrak{f}^{tet}\left(\bar{U}_{c1c}, \bar{U}_{c2c}, \bar{U}_{c3c}, \bar{U}_{c4c}, \bar{T}_{t1t}, \bar{T}_{t2t}, \bar{T}_{t3t}, \bar{T}_{t4t}\right) \tag{3.29}
$$

where

$$
\begin{aligned}
\mathfrak{f}^{tet} &= \text{is the solution to Equation 3.27, chosen via Equation 3.28,} \\
\bar{T}_{tct} &= \text{the translation of the camera relative to the tetrahedron,} \\
\mathbf{R}_{ct} &= \text{the rotation matrix converting from camera to tetrahedron CFs,} \\
\bar{T}_{tit} &= \text{the translation of the } i^{\text{th}} \text{ point of the tetrahedron in its CF and} \\
\bar{U}_{cic} &= \text{the UV pointing from the camera to point } i \text{ of the tetrahedron.}
\end{aligned}
$$

# Chapter 4

# Preliminary investigative research

This chapter provides the results of some preliminary investigative research undertaken in the early phases of the research. The aim of this chapter is to determine the best methods and practices for further development for the APCCS.

Section 4.1 provides a comparison of polar versus Cartesian lens distortion models. The aim of this section is to determine which kind of distortion modelling is the most accurate. The best performing characterisation found in literature [16, 17] is a polar model and it provides a comparison to other models both polar and Cartesian. This section then uses ANNs to act as improved Cartesian models and compares these to models found in literature. This ANN implementation was performed in two phases: an feasibility study followed by an optimisation phase.

Section 4.2 compares different calibration methods, different calibration targets, and different methods to locate the targets with sub-pixel accuracy. A common camera exhibiting significant distortion was used by all calibration methods and targets. The results of all these calibrations are assessed using a common database to determine the accuracy of monocular triangulation. This assessment is then used to determine which calibration method, target and image processing routines should be used for the APCCS.

## 4.1 Comparison of polar and Cartesian models

The aim of this section is to evaluate the relative effectiveness of polar and Cartesian modelling methods and select one for further development and use in the APCCS. ANNs were selected to model the distortion in the Cartesian domain due to their detail-agnostic properties. The ANNs were trained to perform UD mapping using the by-product matched pairs of distorted and undistorted image coordinates created during the DU characterisation. This latter characterisation was performed using the methods described de Villiers et al. [16, 17]. This investigation was performed in two stages: first an initial investigation was performed to assess the feasibility of ANNs for UD modelling. The results of this investigation were published [10] in 2010. The second phase entailed the optimisation of ANNs to determine how accurately they could be made to emulate the UD correction. The results were presented [11] in 2011. These papers may be consulted for details. What follows is a summary of the results and methods that are pertinent to this work. No libraries were used for this work, all the code was developed from first principles.

### 4.1.1 Investigation into artificial neural networks

This work looked at the feasibility of ANNs for UD modelling. ANNs have been used for camera calibration before: Memon and Khan [75] trained an ANN to triangulate the 3D position from matched stereo pair coordinates of a point. Using ANNs, Do [76] modelled both a complete camera system as well as its deviation from the pinhole model. Ahmed et al. [77] modelled the intrinsic and extrinsic calibration parameters of a camera using an ANN, with the intentional exclusion of lens distortion effects.

The following are key points for the initial ANN distortion modelling characterisation:

1. An Allied Vision GE1600 monochrome camera with an 82° Horizontal Field of View (HFOV) 4.8 mm Schneider Cinegon lens was used. This provided severely distorted images as seen in Figure 4.1(a). The initial distortion was 16.7 pixels Root Mean Square (RMS) over the whole image.
2. The work of de Villiers et al. [16,17] was used to perform the DU calibration. The results of the DU calibration were used as training data for the ANNs to perform the UD calibration.
3. A standard fully connected feed forward layered architecture was used for the ANNs.
4. The standard logistic function was used for the neuron activation function.
5. Backwards propagation [78] was used to train the networks.
6. Separate networks (each dependent on both the horizontal and vertical coordinates) were trained to correct either the horizontal or vertical pixel ordinates.
7. Differing numbers of Hidden Layer (HL) together with varying numbers of neurons in each layer were tested.
8. The network architectures with the best average results had their weights numerically refined using the LFA (Section 3.2.2.4) to optimise the RMS results for the entire epoch (rather than considering only one Input/Output (IO) pair at a time).
9. The quantification of line straightness was expressed in terms of microns on the CCD, rather than in pixels to aid comparison to results in literature in a fair and resolution agnostic manner.

The results are provided in Table 4.1 (adapted from the paper the author published [10]). In order to provide context, the results are listed together with results of de Villiers et al. [16] and Mallon and Whelan [44]. Mallon and Whelan are one of the few UD works that provides sufficient information to convert results from the pixel to micron domains. It can be seen that the locally optimised ANNs performed comparably well to results in literature and thus merited further investigation.

Table 4.1: Initial ANN distortion results comparison.

| **Correction** | **Resolution** | | **Pixel** | **Pixel** | **Micron** |
|---|---|---|---|---|---|
| **type** | **Horiz** | **Vert** | **size ($\mu$)** | **error** | **error** |
| ANN 3 HL | 1600 | 1200 | 5.5 | 6.78 | 37.3 |
| ANN 2 HL | 1600 | 1200 | 5.5 | 4.80 | 24.75 |
| ANN LFA optimised | 1600 | 1200 | 5.5 | 0.85 | 4.68 |
| de Villiers et al. [16] | 1600 | 1200 | 5.5 | 0.30 | 1.65 |
| Mallon and Whelan [44] | 1024 | 768 | 8.6 | 0.42 | 3.61 |
| de Villiers et al. [16] | 667 | 502 | 13.15 | 0.013 | 0.17 |

### 4.1.2 Optimisation of artificial neural networks for undistorted to distorted domain mapping

The initial ANN distortion modelling study (Section 4.1.1) showed promising results and suggested some avenues for further improvement. The primary finding of vastly improved results being obtained with refinement of the weights via epoch wide numerical optimisation was carried over. The following additional work was carried out using the same hardware:

1. The output scaling factor of the ANNs was changed so that only a subset of the activation function's finite range mapped on to the image. This allowed the ANNs to not force theoretical pixels falling outside of the camera's FOV to be mapped onto the image.
2. ANNs with three HLs were shown to perform worse (Section 4.1.1) than two HL ANNs, ergo only single and dual HL ANNs were evaluated.
3. The number of neurons per layer varied around the the optimal range observed in the initial ANN investigation (Section 4.1.1).
4. In addition to the original logistic activation function, two different neuron activation functions were investigated: hyperbolic tangent and arctangent.
5. ANN training speeds were improved by comparing single and dual precision floating-point Central Processing Unit (CPU) processing, multi-threading CPU processing and porting the epoch wide fitness evaluation to the Graphics Processing Unit (GPU).
6. An investigation into having a single combined network for correcting both

image coordinates versus separate networks for each image ordinate was done.

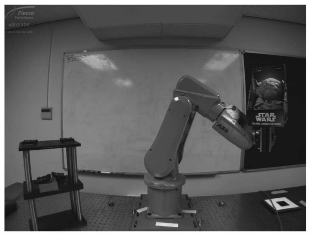
### 4.1.3 Polar versus Cartesian conclusion

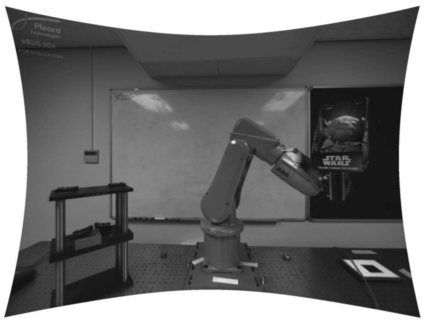
It was seen that the single precision results were noticeably inferior to the dual precision results. Significant performance gains were achieved by multi-threading and porting the training algorithms to the GPU. Having a single network correct both the horizontal and vertical offsets provided better results than two networks each only performing one correction. This was true for both single and dual HL ANNs. The dual layer ANNs took longer to train but yielded superior results. The de facto logistic function was the worst performing of the activation functions. The hyperbolic tangent function provided the best results on average while arctangent yielded the best outright results. Conversely, for the single HL ANNs, the logistic function was by far the best activation function. The final results of the comparison, adapted from the published paper [11], are presented in Table 4.2. Figure 4.1 helps place the numerical results of Table 4.2 in context by displaying distortion correct images obtained using the best performing ANNs. The images in Figure 4.1 are all to the correct scale compared to the original input image of Figure 4.1(a).

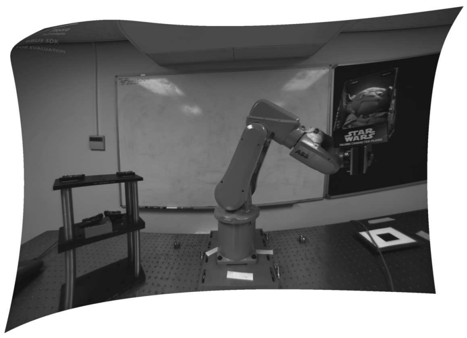
The final results of the ANN investigation show that ANNs achieve results as good as any available in literature [11]. The ANNs also require less expert intervention once an architecture and activation function have been selected: only the IO scaling factors need to be altered for the resolution of the image. However, ANNs require many more parameters: the best performing ANN had 2 HLs of 8 neurons resulting in 116 weights and thresholds. This equates to 20 arctangent function calls, 120 additions/subtractions and 4 multiplications per pixel being evaluated. By comparison, the 5 radial 3 tangential Brown model used by de Villiers et al. [16] requires only in the order of 30 multiplications and 20 additions per pixel. Table 4.3 shows the full screen frame rates for correcting a $1600 \times 1200$ input image resulting in a $2600 \times 2200$ image. It can be seen that ANNs were slower than Brown's model. This is due to the memory architecture

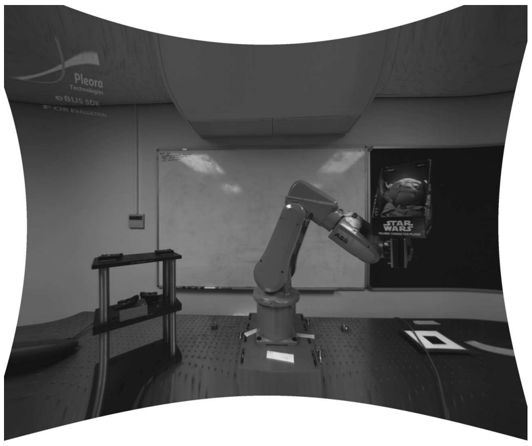
Table 4.2: Comparison of optimised ANN distortion results.

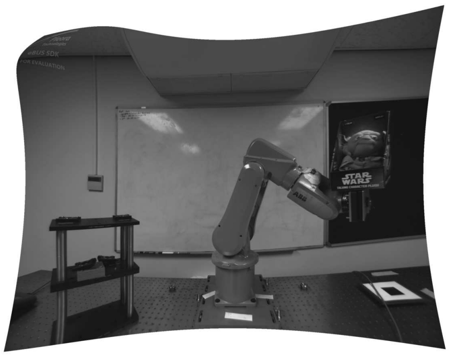
| **Correction type** | **Hidden layers** | **Resolution** | | **Pixel size ($\mu$)** | **Pixel error** | **Micron error** |
|---|---|---|---|---|---|---|
| | | **Horiz** | **Vert** | | | |
| 2 single-output logistic ANNs | 1 | 1600 | 1200 | 5.5 | 1.07 | 5.89 |
| 2 single-output arctan ANNs | 1 | 1600 | 1200 | 5.5 | 28.3 | 155 |
| 2 single-output tanh ANNs | 1 | 1600 | 1200 | 5.5 | 2.83 | 15.6 |
| 1 dual-output logistic ANN | 1 | 1600 | 1200 | 5.5 | 0.73 | 3.91 |
| 1 dual-output arctan ANN | 1 | 1600 | 1200 | 5.5 | 3.70 | 20.4 |
| 1 dual-output tanh ANN | 1 | 1600 | 1200 | 5.5 | 2.02 | 11.1 |
| 2 single-output logistic ANNs | 2 | 1600 | 1200 | 5.5 | 0.60 | 3.30 |
| 2 single-output arctan ANNs | 2 | 1600 | 1200 | 5.5 | 0.59 | 3.23 |
| 2 single-output tanh ANNs | 2 | 1600 | 1200 | 5.5 | 0.45 | 2.48 |
| 1 dual-output logistic ANN | 2 | 1600 | 1200 | 5.5 | 0.51 | 2.81 |
| 1 dual-output arctan ANN | 2 | 1600 | 1200 | 5.5 | 0.31 | 1.71 |
| 1 dual-output tanh ANN | 2 | 1600 | 1200 | 5.5 | 0.44 | 2.42 |
| de Villiers and Nicolls [10] | 2 | 1600 | 1200 | 5.5 | 0.85 | 4.68 |
| de Villiers et al. [16] | N/A | 1600 | 1200 | 5.5 | 0.30 | 1.65 |
| Mallon and Whelan [44] | N/A | 1024 | 768 | 8.6 | 0.42 | 3.61 |
| de Villiers et al. [16] | N/A | 667 | 502 | 13.15 | 0.013 | 0.17 |


(a) Original distorted image.


(b) Image undistorted as per de Villiers et al. [16].


(c) Two pixel error ANN corrected image.


(d) Best initial ANN (4.68 pixel error).


(e) Best single HL ANN (0.57 pixel error).

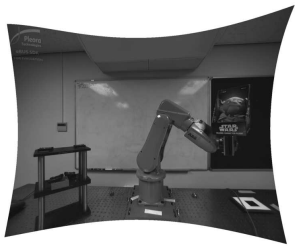

(f) Best dual HL ANN (0.31 pixel error).

Figure 4.1: ANN distortion corrected images.

of the GPUs.

Table 4.3: ANN versus Brown model frame rates.

| **Correction type** | **NVidia GPU used** | | |
|---|---|---|---|
| | **GTX460M** | **NVS5200M** | **GTX560** |
| ANN map | 199 | 85 | 420 |
| Brown, 7 parameters | 230 | 136 | 480 |
| Brown, 10 parameters | 225 | 145 | 500 |

Due to the marginally better performance provided by polar models in addition to their lower processing requirements, it was decided that the APCCS would use polar distortion models.

## 4.2 Comparison of lens calibration accuracies

The aim of this investigative comparison is to determine which distortion calibration methods yield the best accuracies for real-world photogrammetric applications. The assessment considered the following aspects:

1. Different calibration patterns are compared.
2. Different image processing methods to accurately locate the calibration pattern's fiducial points in the images are assessed.
3. Different lens distortion correction models are investigated.

This comparison was presented as a paper [5] in 2011. The comparison is summarised here as justification for the decision to use the work of de Villiers et al. [16] as the basis for the APCCS.

### 4.2.1 Purpose of comparison

Several methods exist for lens distortion calibration and they use a variety of calibration targets and methods to quantify the efficacy of their calibration. It is therefore not always obvious how to compare the calibration fitness of different calibration methods. This comparison aimed to use a single comparison method to compare calibration results. Specifically, the accuracy of displacements measured between high contrast targets on a planar surface that was obliquely observed with a single camera were compared. These measurements are then compared to ground truth displacements that were physically measured.

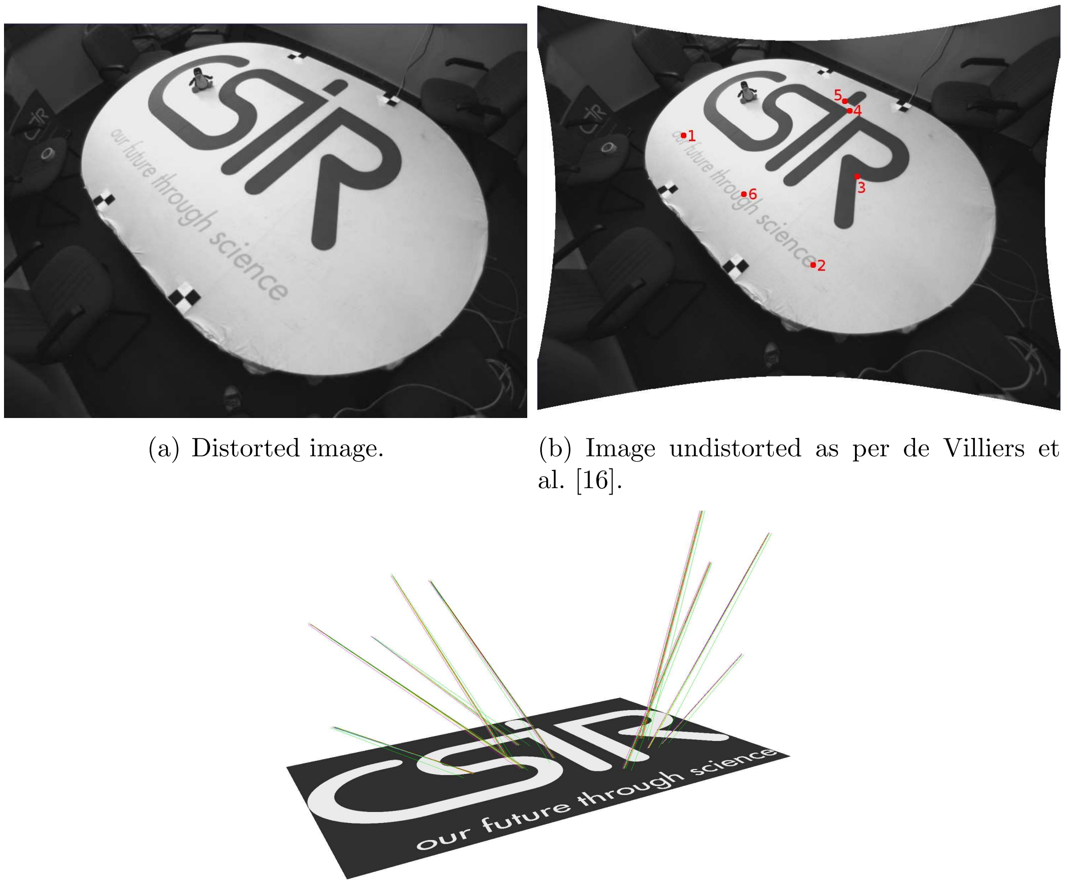
### 4.2.2 Comparison methodology

A single camera was used to create a common data set. Multiple views of a planar pattern were captured and 10 feature points located by hand. Thereafter the camera was calibrated according to all the different methods considered and the common hand captured image feature coordinates used for evaluation. Figure 4.2 shows the planar reference pattern that were used. Figure 4.2(a) is an example of one of the input images used. Figure 4.2(b) shows the same image but undistorted and with the 6 feature points used for displacement measures indicated.

Four checker intersections (Figure 4.2) were used to determine the pose of the camera relative to the plane. Details of this pose determination are given in Section 5.5.4. In total 9 images were used. Figure 4.2(c) shows the calculated positions of the cameras and where their optical axes would have intersected the plane.

Three distortion calibrations were performed: firstly the Brown lens model [34] using five radial, three tangential parameters and an optimal distortion centre was evaluated. A second Brown model with three radial, two tangential parameters and an optimal distortion centre was also evaluated. The methods suggested by de Villiers et al. [16] were used to fit these parameters. Finally, the popular OpenCV Toolkit [4] was used and calibrations performed according to stated best practices.

For the Brown calibrations an LCD was used to display either a single checker intersection or single circle. This fiducial marker was then subsequently moved



(a) Distorted image.

(b) Image undistorted as per de Villiers et al. [16].

(c) Calculated camera positions relative to planar reference.

Figure 4.2: Planar reference pattern, with reference points marked.

in a regular grid such that over time a dense sampling of the entire FOV of the camera was obtained. Two sizes of circle and three sizes of checkers were tested. The circle centres were found using two methods (both described in Section 3.4.5). Either the centroid or the centre of a fitted ellipse was used. The checker intersections were found using the methods of either Luchesse and Mira (LM) [46] or Chen and Zhang (CZ) [47].

Using the intrinsic parameters returned by the calibrations, unit vectors to the four checkers and six logo points were calculated as per Equation 3.17. The four checker vectors were used to determine the pose of the camera relative to the logo plane as per Section 5.5.4. The vectors to the logo points were then extended until they intersected the plane and the distances between these vector-plane intersection points compared to the ground truth. Please refer to the resultant paper [5] for mathematical details of how this was done.

### 4.2.3 Comparison results

Table 4.4, adapted from the work of de Villiers [5], provides the residual distortion calibration results. The results provided for the Brown models are as per Equation 5.6 in Section 5.5.1.

Table 4.4: Distortion metrics.

| Pattern type | Measurement method | Initial distortion (pixels RMS) | Optimal distortion (pixels RMS) |
|---|---|---|---|
| OpenCV calibration[1] | | - | 0.770 |
| Circle, size 10 | Centroid | 347.645 | 0.081 |
| | Ellipse | 347.785 | 0.088 |
| Circle, size 25 | Centroid | 335.622 | 0.078 |
| | Ellipse | 335.510 | 0.142 |
| Square, size 15 | LM[2] | 386.059 | 0.256 |
| | CZ [3] | 340.954 | 0.082 |
| Square, size 25 | LM[2] | 386.342 | 0.103 |
| | CZ [3] | 314.710 | 0.081 |
| Square, size 50 | LM[2] | 386.695 | 0.099 |
| | CZ [3] | 251.682 | 0.060 |

[1] Open Computer Vision [4] reprojection error, not Equation 5.6.
[2] Luchesse and Mira [46]
[3] Chen and Zhang [47]

Table 4.5 provides results for the single camera photogrammetric displacement measurements in the planar surface. The value provided is the RMS error in millimetres. Both the global results and the results for only the first 5 images are provided. This is because the first images were less oblique (with regard to the plane normal vector) and had noticeably different results due to better localisation of all of the feature points.

Table 4.5: RMS 3D error resulting from camera calibration patterns.

| Pattern type | Measurement method | Images 1-5 error (mm) | Global error (mm) |
|---|---|---|---|
| OpenCV Calibration[1] | | 3.66 | 4.58 |
| Circle, size 10 | Centroid | 3.36 | 4.29 |
| | Ellipse | 3.26 | 4.28 |
| Circle, size 25 | Centroid | 3.31 | 4.84 |
| | Ellipse | 3.02 | 3.95 |
| Square, size 15 | LM[2] | 2.84 | 4.11 |
| | CZ [3] | 3.13 | 3.98 |
| Square, size 25 | LM[2] | 3.05 | 3.91 |
| | CZ [3] | 3.03 | 3.83 |
| Square, size 50 | LM[2] | 3.08 | 4.01 |
| | CZ [3] | 3.11 | 3.88 |

[1] Open Computer Vision Tool Box [4].

[2] Luchesse and Mira [46].

[3] Chen and Zhang [47].

### 4.2.4 Comparison conclusion

From Table 4.5 it can be seen that the ellipse centre provides better results than the centroid when using a circle pattern. Chen and Zhang's method [47] yields superior results to Lucchesse and Mira's method [46]. All of the circle and square calibrations used the work of de Villiers et al. [16]. In general, these methods all outperformed the OpenCV calibration. Thus it was decided to use the method of de Villiers et al. [16] for the rest of this work.

# Chapter 5

# Calibration routines

This chapter addresses the first and primary research question posed in Section 1.2: *"Can an automated photogrammetric camera calibration system meeting the criteria listed in Section 1.1.3 be created using a robotic arm?"* This chapter describes the algorithms developed for the APCCS, and constitutes the majority of the novelty. By the time of submission the core idea of using a robot arm for the calibration was no longer novel. However, the combinations of capabilities (specifically the ability to calibrate a single camera) and several of the algorithms to characterise specific parameters are novel. These algorithms are identified in their descriptions.

Section 5.1 provides a detailed description of the APCCS. Essentially, the APCCS consists of a robotic arm (an ABB IRB120) mounted on an optical table which has some mounting points for cameras. A light source is attached to the end effector of the robot. A Personal Computer (PC) controls both the arm and the camera being calibrated.

A high level functional overview of the APCCS if given in Section 5.2. The PC causes the camera to take an image of the light source and records the image coordinates of the light source and pose of the robot arm. The robot movement sequence is unique for each photogrammetric parameter. Processing the captured data after the movement sequence is completed results in the photogrammetric parameter currently being calibrated. Section 5.2 also provides a thorough description of the dependencies of the calibrations on each other and provides the correct order in which the calibrations must be performed.

Conceptually one may consider placing the camera on the robot arm and having it observe a stationary light source. Section 5.3 discusses this in more detail as well as the advantages and disadvantages of each configuration.

In total 6 calibrations are described in this chapter: 2 are required to calibrate the APCCS itself and 4 are to calibrate the cameras. All the calibrations are optical requiring no external equipment, with the exception of the calibration of the PMJ mounting brackets. This complex and intricate set of calibration processes is necessary to achieve all the goals of the APCCS. Specifically the complexity is necessary to calibrate cameras of any spectrum and FOV. Calibrating in different spectrums requires changing the light source, which means the position of the light source w.r.t. the end effector can change. This is why the LED offset needs to be determined. The rest of the complexity is to allow for the mounts on which the cameras are placed to be moved to cater for different FOVs. None of the calibrations make use of a known mount pose w.r.t the robot, this greatly complicates the calibration procedures. The additional benefit of this complexity is that a new APCCS can be unpacked and configured rapidly as only rough measurements are required to ensure end effector is in the camera's FOV.

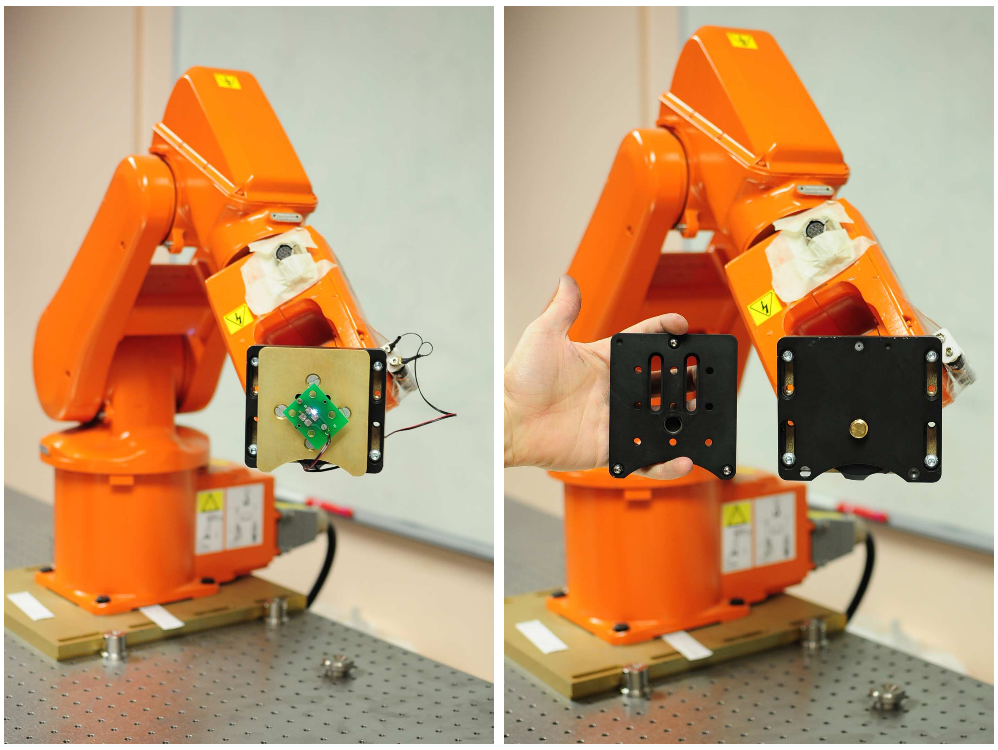
Section 5.4 discusses the two routines necessary for calibration of the APCCS itself. Section 5.5 provides the mathematical details of the routines to photogrammetrically calibrate a camera once the necessary APCCS calibrations have been performed.

## 5.1 Equipment description

This section provides a detailed description of the apparatus that constitutes the APCCS. The APCCS uses a robotic arm to place a LED at a sequence of known positions, relative to the camera being calibrated. The movement sequence can be updated to better measure different photogrammetric parameters or for changes in camera FOV. The absolute pose of the camera during calibration is not required, but it is required to be stable and repeatable. The complete calibration of a camera may require it to observe several different movement sequences from several different vantage points.

The cornerstone of the APCCS is a robotic arm. The arm used has 6 rotation joints which together allow for it to place the end of the arm (also called the 'end effector') at specified poses. The full 6 DOF pose (i.e. translation and orientation) of the end of the arm can be specified. Of course there are physical limitations as to where the end of the arm can be placed, but it is normally feasible to find a camera mounting pose and movement sequence that falls within the movement envelope of the robot. The robot outputs the pose of the end effector, not the tip of any devices attached to the end effector. In this work an ABB IRB120 robot and an IRC5 compact controller were used which together have a claimed accuracy of 10 microns. The angular output format of the ABB IRB120 is unusual and undocumented. Section 3.3.4 describes how the angles received from IRB120 are converted to the robot CF. Figure 5.1(a) shows the ABB IRB120 robot arm used.

The robot arm is used as a repeatable and configurable LED placement device (Figure 5.1(a)). The LED is attached to the end of this arm via a repeatable kinematic mount (Figure 5.1(b)). The mount allows the LED to be removed and replaced without redetermining the spatial offset of the LED (Section 5.4.2). The LED may be swapped with a different LED, to then calibrate a camera with a different sensitivity spectrum (e.g. calibrating a visual camera followed by calibrating a LWIR camera). The maximum extent of the envelope through which the robot arm is moved can be altered to accommodate cameras of differing FOVs that are being calibrated. The density of the sampling within the envelope can be altered to trade off calibration accuracy versus calibration time. Figure 5.1(b) shows the Newport M-BKL-4 kinematic mechanically locking mounting interface used on the robot for LED and helmet mounting. This mount has a stated repeatability of better than 100 microradians [79].



(a) Robot arm with LED. (b) Robot arm with kinematic mount.

Figure 5.1: ABB IRB120 robot arm.

The robot is placed on an optical table. In this work a 1.8 m by 2.4 m Integrity 3 VTS 12 inch table was used to minimise vibration effects of the moving arm on the rest of the table. It also facilitated the convenient addition of mounting brackets from which cameras can be calibrated. Figure 5.2 shows the table with the robot, two camera mounting positions, and the PMJ.

The robot and optical table are placed in a temperature controlled room which has had all of its windows boarded. The temperature control minimises the effects of thermal variance during calibration. The light control allows long exposures without incurring and stray light effects. The robot is programmed to pause at each of the discrete locations until one or more images have successfully been calibrated. These last two characteristics make the system resilient to different camera integration and read-out strategies such as rolling versus global shutters, and interlaced versus non-interlaced readouts. The immobility of the robot arm

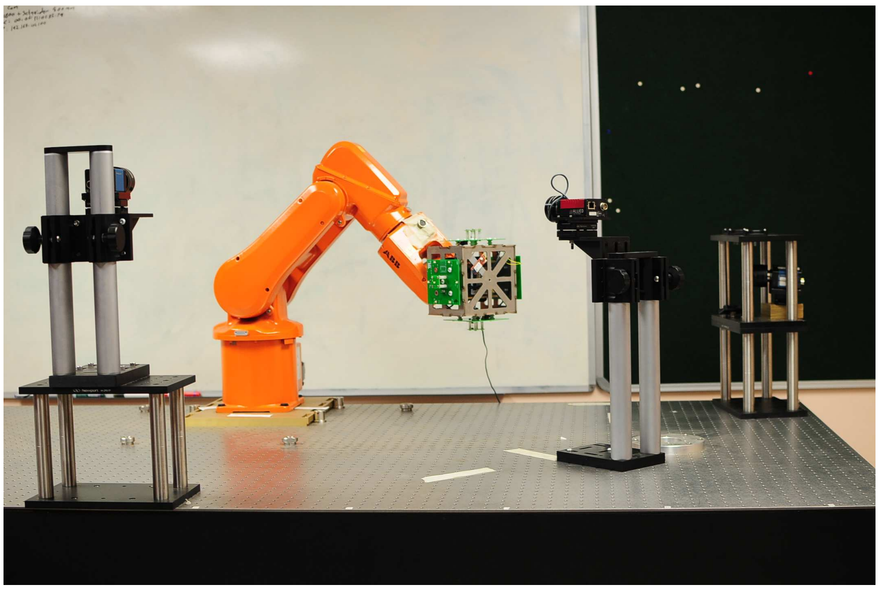


Figure 5.2: Newport Integrity 3 VCS optical table.

during image capture ensures there is no motion blur or tearing of the LED in the camera images.

Cameras to be calibrated are placed on kinematic mounting locations on the table. Figure 5.3 shows the Newport M-BK-2A magnetically locating mount that was used. These mounts also have a stated repeatability of less than 100 microradians [79]. Placing the cameras on repeatable mounts allows the extrinsic parameters to be split into two components. The split extrinsic pose allows faulty cameras in an end user system to be replaced without repeating a calibration of the entire end user system.

Some of the calibrations require the use of a PMJ. The PMJ is a removable set of repeatable kinematic camera mounting interfaces whose poses are known relative to each other. The PMJ, shown in Figure 5.4, is temporarily mated to the optical table when required and removed when no longer needed. All the calibrations that make use of the PMJ do not require knowledge of its pose relative to the robot arm, merely knowledge of the relative poses of its mounts.

The optical table of a deployed APCCS will have a robotic arm with an LED attached via a kinematic mount. This LED is observed by the camera being

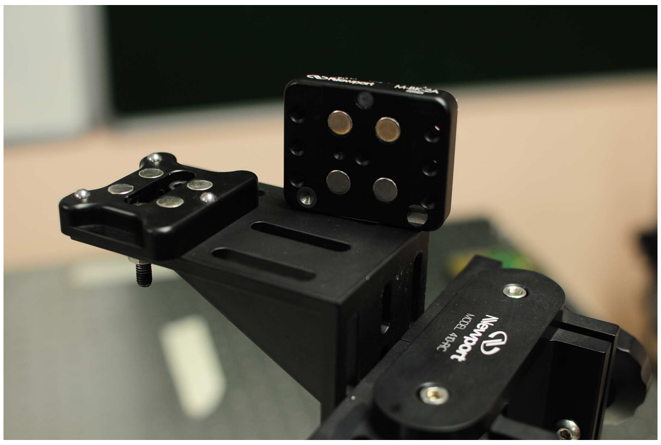


Figure 5.3: Close-up of Newport M-BK-2A.

calibrated, which is mounted on a repeatable kinematic mounting bracket. This bracket is either part of a permanently attached mounting point or the temporarily affixed PMJ.

The final component of the APCCS is the PC which controls the data acquisition, processing and calibrations. The primary functions of the computer are to:

1. Command the robot to the next pose in the sequence and record its exact resultant pose.
2. Capture the camera's image of the LED at the current robot arm pose.
3. Process the captured image to determine the centre of the LED.
4. Process the captured LED centres, together with the corresponding actual robot poses to determine the camera calibration parameters.
5. Store and report the results of the camera calibrations.

## 5.2 Functional overview

This section describes the procedure that is followed to calibrate a camera. Two sets of calibration parameters are desired for a camera: intrinsic parameters and extrinsic parameters. The intrinsic parameters allow image coordinates to be converted into 3D vectors in the camera's CF and 3D points expressed in the

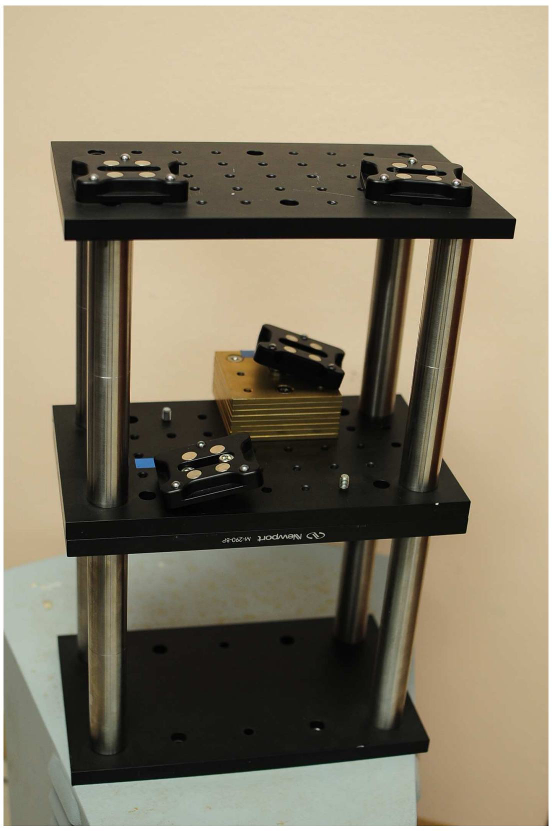


Figure 5.4: Precision mount jig.

camera's CF to be mapped onto their corresponding image coordinates. This consists of the DU parameters, the UD parameters and the pinhole equivalent focal length. It assumed the pixel dimensions and the skewness (non-orthogonality of CCD horizontal and vertical axes) can be obtained from the data sheet.

The extrinsic parameters consist of the camera's pose relative to a chosen reference or world CF, allowing translation between the camera's CF and the reference CF. For operational ease of use this pose is split into two separate poses, the pose of the camera w.r.t. a repeatable mechanical mount, and the pose of the repeatable mechanical mount w.r.t. the reference CF. This allows a camera to be replaced in a system without repeating the calibration of the mount poses.

Once the APCCS has been configured, the following calibrations are performed per camera:

1. The camera captures centroids of the robot arm moving the LED through a large regular planar grid that subtends the camera's entire FOV.
2. Analysis of the LED centroids is performed to determine the DU parameters (Section 5.5.1).
3. Processing of the corresponding distorted and undistorted pixel coordinates generated during the DU parameter determination is performed to yield the UD parameters (Section 5.5.2).
4. The camera captures centroids of the robot arm moving the LED to the vertices of several *a priori* known tetrahedrons.
5. Processing of the LED centroids, combined with the DU parameters and robot poses results in the pinhole equivalent focal length (Section 5.5.3).
6. The PMJ is affixed to the table. For each of its mounting points the camera captures centroids of the robot arm moving the LED through the same series of 3D grids.
7. Simultaneous analysis of each of the set of centroids captured from each of the PMJ's mounts, combined with the DU parameters, focal length, and known relative poses of the PMJ mounts yields the (irrelevant) PMJ-to-robot pose as well as the (desired) camera-to-mount pose (Section 5.5.5).
8. The camera is placed on the operational mount and the camera captures the centroids of the robot arm moving the LED through a series of planar grids.
9. The centroids, together with the robot positions using the DU parameters, focal length, and LED-to-arm offset are processed to determine the pose of the camera relative to the robot arm. The pose of the mount on which the camera is mounted is calculated from the camera-to-robot pose and the camera-to-mount pose (Section 5.5.4).

The preceding list requires certain calibrations of the APCCS to have been performed. Specifically, the relative poses of the PMJ's mounts are required - these are obtained via external measurements described in Section 5.4.1. Similarly, the LED spatial offset relative to the robot arm is required. This offset is measured optically and requires the PMJ to have been calibrated and a camera to have been calibrated to determine its DU parameters, focal length and camera-to-mount

Table 5.1: Calibration dependencies.

| **Calibration** | | **Dependencies** | | | | | |
|---|---|---|---|---|---|---|---|
| **Number** | **Description** | **1** | **2** | **3** | **4** | **5** | **6** |
| 1 | Lens distortion correction | - | E | E | E | E | - |
| 2 | Focal length determination | R | - | E | E | E | - |
| 3 | Camera mount pose offset determination | R | R | - | E | E | R |
| 4 | Mount pose determination | R | R | R | - | R | R* |
| 5 | LED translation offset | R | R | R | E | - | R |
| 6 | Precision mounting jig calibration | - | - | E | E* | E | - |

Legend:

| | |
|---|---|
| - | = No relation. |
| E | = Directly enables. |
| E* | = Indirectly enables, i.e. enables a calibration that this calibration requires. |
| R | = Directly requires. |
| R* | = Indirectly requires, i.e. requires a calibration which in turn requires this calibration. |

pose. Section 5.4.2 describes how the LED spatial offset is measured. Note that neither the PMJ calibration nor the LED spatial offset are required to be performed for each camera, merely initially and then (potentially) at predetermined set intervals.

Table 5.1 lists the calibrations just discussed and indicates the dependencies of each calibration on the others.

## 5.3 Close range versus goniometric calibration

The calibration methods outlined in Section 5.2 and detailed in the rest of this chapter are close range calibration methods, meaning that the cameras need to be placed close the robot arm in order to have their full FOV covered. This means that the camera needs to be focussed on the robot and so brings in to question the applicability of these calibrations for long range photogrammetric calibrations. An alternative to close range calibration is the goniometer style of calibration. Clarke and Fryer [80] provide a comparison of these two families of camera calibration methods.

Goniometer calibrations place the camera on the moveable component and observe a stationary light source, typically through a collimator so the camera can be focussed at infinity. This infinity focus makes the calibrations easier to apply to long range applications. For a robotic arm based system there are several disadvantages to such a set up:

1. Collimators typically have a narrow spectrum over which they operate, e.g. visual collimators are opaque at LWIR wavelengths. Several collimators would then be required.
2. Only cameras that fit within the mass and torque specification of the robot arm may be calibrated, potentially eliminating heavy cameras such as cooled IR cameras.
3. Similar to the above, the volume of the cameras can diminish the movement envelope of the robot arm. This is because the camera may interfere with the arm when the arm is in certain orientations. This effect is worsened if an array of cameras has to be calibrated simultaneously.
4. The robot arm needs to facilitate the transport of the camera images to the PC for processing. This entails either (expensive) high bandwidth slip rings, or (potentially bulky) looms that have to be fixed to the arm, and could thus hinder the robot arm's movement and accuracy.
5. Providing power, sychronisation and control signals to the camera are similarly made more difficult when the camera is mounted on the robot arm.
6. If the robot arm experiences a movement error, the camera may be damaged. Light sources are typically less expensive than cameras of the same wavelength.

In the APCCS to partially mitigate the effect of calibrating at close range, the camera is focussed at the application's operational range and the iris is stopped down to bring the light source closer to focus. Here it is worth noting that stopping down the iris causes the integration time to increase, but so would a collimator's non-perfect optical transmission. Also to be noted is that it is preferable to have the image of the LED not be perfectly focussed as this spreads the energy over a few pixels. This spreading allowing the centroid and ellipse fitting methods described in Section 3.4.5 to partially mitigate the effects of discrete spatial sampling, quantisation and non 100% fill factor of the CCD. Section 6.1 uses cameras calibrated by the APCCS in this manner in an outdoor stitching application. The area of interest is effectively at infinity for the short focal length cameras that are used. The favourable results of Section 6.1 provide some evidence that the APCCS calibrations are applicable to both long and short range applications.

## 5.4 Calibration of the table

This section describes the procedures necessary to calibrate components of the APCCS so that it can then be used for camera calibration. Section 5.4.1 describes the calibration of the PMJ, and Section 5.4.2 describes the determination of the 3D spatial offset of the LED relative to the end effector.

### 5.4.1 Precision mounting jig calibration

This section details the mathematics used to calibrate the PMJ. The PMJ is required to determine the pose of the camera relative to its mounting interface (Section 5.5.4). The top two mounts of the jig are also used to create a stereo pair to determine the spatial offset of the LED relative to the robot arm (Section 5.4.2). A Computer Aided Design (CAD) model of the PMJ is shown in Figure 5.5.

The PMJ is made up of four Newport M-BK-2A kinematic mounts. Each interface has three Ball Bearing (BB)s used for the repeatable mounting. The kinematic mounts are labelled A to D, and the BBs are labelled 1 to 3 (Figure 5.5). The purpose of the methods in this section is to find the positions of each kinematic mount relative to mount A. In all of the calculations, BB 1 of each

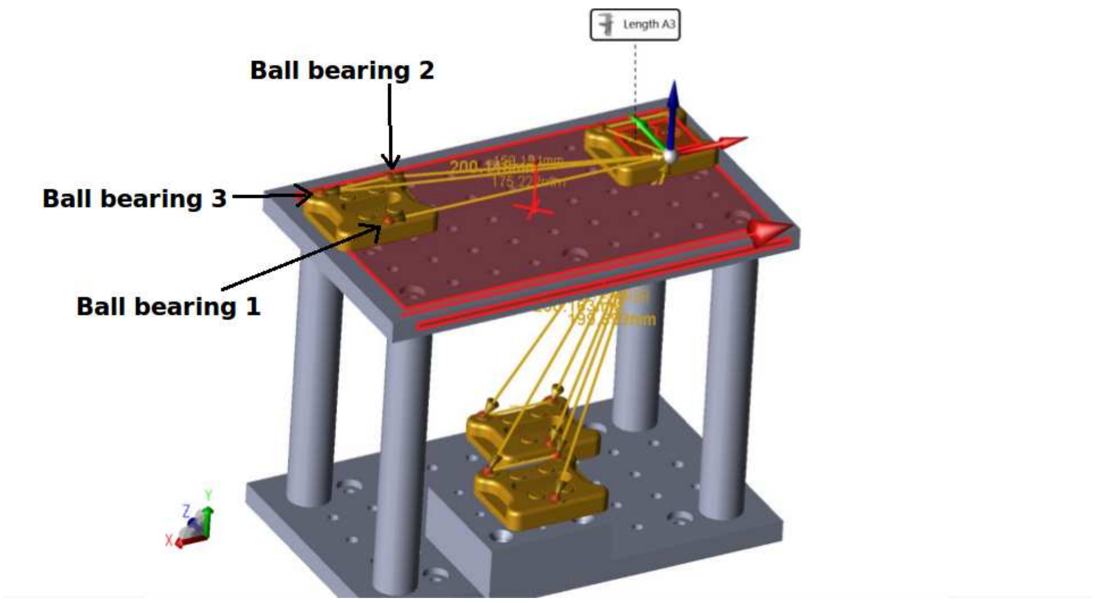


Figure 5.5: CAD model of the PMJ.

mount is used as the origin of that mount's CF.

Using a high accuracy mechanical measurement station (in this work a Faro Arm Platinum portable contact measurement machine with a 1.2 m span) the positions of each of the BBs on all of the mounts were measured. All of the measurements made by the Faro arm were relative to its CF. It is therefore necessary to change the measurements so they are in the CF system defined by Figure 3.1. To do this, the poses of all four mounts relative to the Faro arm were calculated with Equation 5.1. The poses of the mounts relative to the first mount were then determined.

Note that it is critical that the mounts do not all have their CFs parallel: at least two axes must be misaligned over the four mounts. This is to avoid ill-conditioned matrices which yield no unique solution when separating the extrinsic parameters into mount w.r.t. the reference and camera w.r.t. the mount. In this work, the top two mounts are only misaligned in yaw (looking inwards) and the bottom two in yaw and roll. This allows the top two to be used as a stereo pair. For each mount the direction of its $X$ axis is defined as from BB1 to the midpoint between BB2 and BB3. The vector from BB2 and BB3 is used to define the

$XY$ plane of the mount's CF. The vector cross product of these vectors yields the $Z$ axis. To ensure orthogonality, the vector cross product of the $Z$ and $X$ axes yields the final $Y$ axis. These three vectors are normalised and then used to create a rotation matrix which converts from the mount's CF to the Faro arm's CF. Equation 5.1 presents this mathematically:

$$\begin{aligned}
\mathbf{R}_{M_iF} &= \mathfrak{f}^{UVtoR}\left(\bar{U}_{M_iXF}, \bar{U}_{M_iYF}, \bar{U}_{M_iZF}\right), \text{ and} \\
\bar{T}_{FM_iF} &= \bar{T}_{FB_{i,1}F} \\
&\text{where} \\
\mathfrak{f}^{UVtoR} &= \text{as per Equation 3.14}, \\
\bar{T}_{FB_{i,j}F} &= \text{3D position of BB } j \text{ of mount } i, \\
\bar{U}_{M_iXF} &= \text{the } X \text{ axis of mount } i, \\
&= \frac{\frac{1}{2}\left(\bar{T}_{FB_{i,2}F} + \bar{T}_{FB_{i,3}F}\right) - \bar{T}_{FB_{i,1}F}}{\left\|\frac{1}{2}\left(\bar{T}_{FB_{i,2}F} + \bar{T}_{FB_{i,3}F}\right) - \bar{T}_{FB_{i,1}F}\right\|}, \\
\bar{U}'_{M_iYF} &= \text{a vector in mount } i\text{'s } XY \text{ plane}, \\
&= \frac{\bar{T}_{FB_{i,3}F} - \bar{T}_{FB_{i,2}F}}{\left\|\bar{T}_{FB_{i,3}F} - \bar{T}_{FB_{i,2}F}\right\|}, \\
\bar{U}_{M_iZF} &= \text{the } Z \text{ axis of mount } i, \\
&= \frac{\bar{U}_{M_iXF} \otimes \bar{U}'_{M_iYF}}{\left\|\bar{U}_{M_iXF} \otimes \bar{U}'_{M_iYF}\right\|}, \text{ and} \\
\bar{U}_{M_iYF} &= \text{the } Y \text{ axis of mount } i, \\
&= \frac{\bar{U}_{M_iZF} \otimes \bar{U}_{M_iXF}}{\left\|\bar{U}_{M_iZF} \otimes \bar{U}_{M_iXF}\right\|}.
\end{aligned} \tag{5.1}$$

The final desired output is the pose of each of the PMJ's mounts w.r.t. the first mount expressed in the first mount's CF. Equation 5.2 shows how this is done:

$$\begin{aligned}
\mathbf{R}_{M_iM_1} &= \mathbf{R}^T_{M_1F}\mathbf{R}_{M_iF}, \text{ and} \\
\bar{T}_{M_1M_iM_1} &= \mathbf{R}^T_{M_1F}\left(\bar{T}_{FM_iF} - \bar{T}_{FM_1F}\right) \\
&\text{where} \\
\mathbf{R}_{M_iF}, \bar{T}_{FM_iF} &= \text{the pose of mount } i \text{ w.r.t. the Faro arm as per Equation 5.1},
\end{aligned} \tag{5.2}$$

$$\mathbf{R}_{M_iM_1} = \text{the rotation matrix of mount } i \text{ relative to mount 1, and}$$
$$\bar{T}_{M_1M_iM_1} = \text{the translation of mount } i \text{ from mount 1, in mount 1's CF.}$$

### 5.4.2 LED translation offset determination

Knowing the spatial offset of the LED relative to the end effector, allows the actual position of the LED to be used in calculations. Without this knowledge the orientation of the end effector of the arm has to remain constant so that the unknown spatial offset of the LED is a constant vector in the reference frame. This does not affect the accuracy with which a camera observing the LED positions has its pose determined, it merely has a spatial error equal to the unknown LED spatial offset. This is acceptable for lens distortion characterisation (which only corrects for straight line projection), Separation of the Extrinsic Parameters (SEP) characterisation (which looks at the relative poses of the camera when mounted on the PMJ), and focal length determination (which merely looks at the tightness of the locus of determined camera poses, not their absolute positions). All the other calibrations require knowledge of this parameter (Table 5.1).

The LED offset is determined via a stereo observation of the LED while the position of the end effector remains constant, but its orientation is varied such that the energy source moves in a circle. Before this, the orientation of the stereo apparatus relative to the robot is determined.

The PMJ is used to create a temporal stereo camera pair by using the top two mounting points. This is why both the camera's offset w.r.t. mount and the relative pose of the PMJ's mounts are required for this calibration. The PMJ is placed at an approximate position and securely fastened to the table. The camera (already calibrated for lens distortion and focal length) is then placed on the first mounting point and records the LED centroids as it observes the robot's movement sequence. Thereafter, the camera is placed on the second mounting point without disturbing the pose of the PMJ w.r.t. the robot. The camera then captures a second set of centroids as it observes the robot sequence from the new vantage point.

The robot movement sequence is divided into two parts. The first part is used to determine the orientation of the PMJ relative to the robot (Section 5.4.2.1). This is why the PMJ pose needs not be known a priori. The second part is used, in conjunction with the results from the first section, to determine the LED offset (Section 5.4.2.2).

#### 5.4.2.1 Determining stereo pair to robot orientation

After the camera has been placed on the secured PMJ, the robot moves through a sequence of points in the $X$ direction of the robots CF. Thereafter it moves in the $Y$ and then the $Z$ directions. This is repeated once the camera has been placed on the second mount.

Using stereo vision the points along each axis are triangulated in 3D and the best fit UVs (expressed in the stereo CF, which in this case coincides with the PMJ CF) pointing along those lines is determined. Once UVs for each axis have been found Equation 3.14 provides the rotation matrix of the robot relative to the jig, which merely has to be transposed.

The identity $\bar{U} \bullet \bar{V} = \|\bar{U}\|\|\bar{V}\| \cos\theta$ is used to fit a 3D line through a sequence of points. If $\bar{U}$ is a UV then $\bar{U} \bullet \bar{V}$ is maximised when $\bar{U}$ is parallel to $\bar{V}$, in which case $\|\bar{U} \bullet \bar{V}\| = \|\bar{V}\|$. Vectors are constructed based on the difference of the points along the $X$ axis from the first point. A system of simultaneous equations is then constructed to find the best fit UV through these points. This UV is the desired $X$ axis UV of the robot in stereo pair's CF (i.e. $U_{sx_rs}$). It is necessary that the points were captured or subsequently sorted to be in increasing order from the first point, i.e. $\|\bar{T}_{sp_{x,m}s} - \bar{T}_{sp_{x,0}s}\| > \|\bar{T}_{sp_{x,m-1}s} - \bar{T}_{sp_{x,0}s}\| \forall m \in ([2, N) \cap \mathbb{Z})$. Equation 5.3 shows how the desired UV is obtained from the ordered points by analytically finding the RMS best fit unit vector which maximises the length of each vector in the set of simultaneous equations.

$$\bar{U}_{sx_rs} = \frac{\bar{V}_{sx_rs}}{\|\bar{V}_{sx_rs}\|} \tag{5.3}$$

where

$$\begin{aligned}
\bar{V}_{sx_rs} &= \left(\mathbf{A}^T\mathbf{A}\right)^{-1}\mathbf{A}^T\bar{B}, \\
\mathbf{A} &= \begin{bmatrix} \bar{T}^T_{p_{x,0}p_{x,1}s} \\ \bar{T}^T_{p_{x,0}p_{x,2}s} \\ \vdots \\ \bar{T}^T_{p_{x,0}p_{x,N-1}s} \end{bmatrix}, \\
\bar{B} &= \begin{bmatrix} \|\bar{T}_{p_{x,0}p_{x,1}s}\| \\ \|\bar{T}_{p_{x,0}p_{x,2}s}\| \\ \vdots \\ \|\bar{T}_{p_{x,0}p_{x,N-1}s}\| \end{bmatrix}, \\
\bar{T}_{p_{x,0}p_{x,i}s} &= \bar{T}_{sp_{x,i}s} - \bar{T}_{sp_{x,0}s}, \\
\bar{T}_{sp_{x,i}s} &= \mathfrak{f}^{\text{int}}\left(\bar{T}_{M_1M_aM_1}, \mathbf{R}_{M_aM_1}\bar{U}_{C_ap_{x,i}C_a}, \bar{T}_{M_1M_bM_1}, \mathbf{R}_{M_bM_1}\bar{U}_{C_bp_{x,i}C_b}\right), \\
\mathfrak{f}^{\text{int}} &= \text{as per Equation 3.20 of Section 3.4.4}, \\
\bar{U}_{C_ap_{x,i}C_a} &= \mathfrak{f}^{\text{img}\rightarrow\text{vec}}(\mathfrak{f}^{\text{Brown}}(\bar{P}^d_{i,a}, \bar{DU}_{\text{params},a}), \bar{I}_{\text{params},a}), \\
\bar{U}_{C_bp_{x,i}C_b} &= \mathfrak{f}^{\text{img}\rightarrow\text{vec}}(\mathfrak{f}^{\text{Brown}}(\bar{P}^d_{i,b}, \bar{DU}_{\text{params},b}), \bar{I}_{\text{params},b}), \\
\bar{DU}_{\text{params},m} &= \text{camera } m\text{'s distortion parameters}, \\
\bar{I}_{\text{params},m} &= \text{camera } m\text{'s intrinsic parameters}, \\
\mathfrak{f}^{\text{img}\rightarrow\text{vec}} &= \text{as per Equation 3.17}, \\
\mathfrak{f}^{\text{Brown}} &= \text{as per Equation 3.25 (Section 3.4.6 for parameter details)}, \\
\bar{P}^d_{i,j} &= \text{the LED centroid (Section 3.4.5) of point } i \text{ by camera } j, \\
\bar{T}_{M_1M_iM_1} &= \text{translation of mount } i \text{ of the PMJ (Section 5.4.1)}, \\
\mathbf{R}_{M_iM_1} &= \text{orientation of mount } i \text{ of the PMJ (Section 5.4.1)}, \\
a &= \text{the number of the first stereo mount position, and} \\
b &= \text{the number of the second stereo mount position.}
\end{aligned}$$

Equation 5.3 is then repeated using the $Y$ and $Z$ axis points to get $\bar{U}_{sy_rs}$ and $\bar{U}_{sz_rs}$ respectively, which can then collectively be passed to Equation 3.14 to get the transpose of the desired orientation of the stereo pair relative to the robot.

Equation 5.4 displays this with mathematical succinctness:

$$\mathbf{R}_{sr} = \mathbf{R}_{rs}^{T} \tag{5.4}$$

where

$$\begin{aligned}
\mathbf{R}_{rs} &= \mathfrak{f}^{UVtoR}\left(\bar{U}_{sx_rs}, \bar{U}_{sy_rs}, \bar{U}_{sz_rs}\right), \\
\mathfrak{f}^{UVtoR} &= \text{as per Equation 3.14}, \\
\bar{U}_{sx_rs} &= \text{robot's } X \text{ axis (from Equation 5.3) in the PMJ's CF}, \\
\bar{U}_{sy_rs} &= \text{robot's } Y \text{ axis (from Equation 5.3) in the PMJ's CF, and} \\
\bar{U}_{sz_rs} &= \text{robot's } Z \text{ axis (from Equation 5.3) in the PMJ's CF.}
\end{aligned}$$

#### 5.4.2.2 Triangulating the LED spatial offset

In order to calculate the LED spatial offset, the translation of the arm's end effector is kept constant whilst its orientation is changed to cause the energy source to trace a circle in space. Typically the end effector is positioned between the stereo pair, about 0.3m from the baseline and pointed directly towards the centre of the cameras' baseline. The circle thus traced by changing the end effector's yaw and pitch relative to this viewing direction is in a plane roughly parallel to that of the cameras' CCDs. Figure 5.6 shows the captured positions of the LED centroid that were recorded from one of the stereo camera poses. The robot arm is visible behind a Teflon shield used to mask the heat source, except through a small central aperture which is at the top of the captured circle.

At each point the position of the energy source is triangulated. These points are then converted to the robot CF using the axis rotation determined by Equation 5.4. The vectors between all the permutations of the points are then calculated. These vectors and called $\bar{T}^{S}_{p_ip_jr}$ indicating they are the displacements from point $i$ to point $j$ expressed in the robot CF and were determined from stereo vision.

Thereafter, a hypothetical 3D translation of the energy source relative to the robot end effector is chosen. With this offset and the known robot orientations, the hypothetical 3D position of the energy source is calculated for all the orientations and the same vectors between all permutations is then created. These

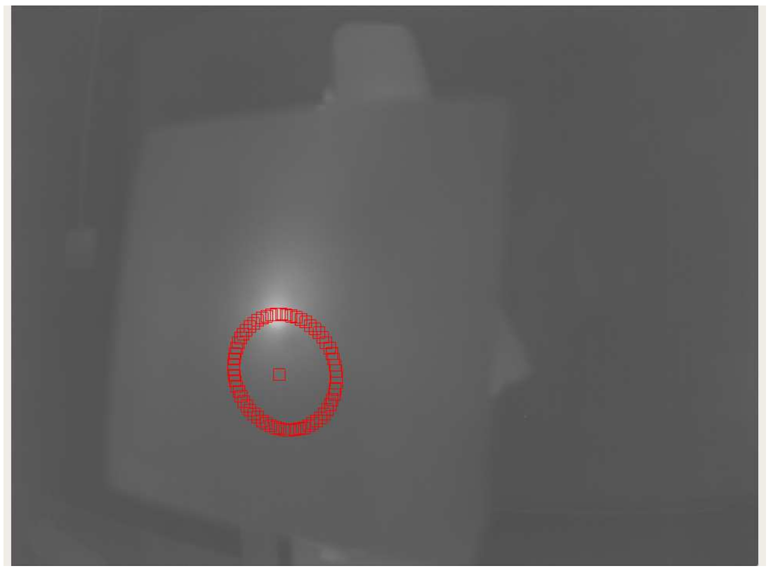


Figure 5.6: Circle of LED positions recorded during a LWIR camera calibration.

vectors and called $\bar{T}^{H}_{p_i p_j r}$ indicating they are the displacements from point $i$ to point $j$ expressed in the robot CF and were determined based on a hypothesised LED offset.

The sum of squares of the magnitude of the difference vectors (between corresponding hypothesis based vectors and stereo vision based vectors) is then determined. This sum will only be zero if the hypothesised LED 3D spatial offset is correct (and the stereo triangulation is accurate), it will be greater than zero for all other LED offsets. Therefore this sum is used as a cost function called $\mathrm{C}^{LP}$, and is minimised via a chosen numerical optimisation technique. Leapfrog [26] was used to perform the minimisation due to its known robustness to noise and ability to find a 'low local minimum'. Refer to Sections 2.2 and 3.2 for the characteristics and mathematical workings respectively of the numeric optimisation routines used. The mathematical derivation of the cost function is given by Equation 5.5:

$$\mathrm{C}^{LP}(\bar{T}^{H}_{ala}) = \sum_{i=0}^{N-2} \sum_{j=i+1}^{N-1} \left\| \bar{T}^{H}_{p_i p_j r} - \bar{T}^{S}_{p_i p_j r} \right\|^2 \tag{5.5}$$

where

$$
\begin{aligned}
\mathrm{C}^{LP} &= \text{the LED position cost function,}\\
\bar{T}^{H}_{ala} &= \text{hypothesised LED translation w.r.t. the end effector}\\
N &= \text{the number of points captured,}\\
\bar{T}^{H}_{p_i p_j r} &= \text{hypothesis based delta vector between points } i \text{ and } j,\\
&= \mathbf{R}_{a_j r}\bar{T}^{H}_{ala} - \mathbf{R}_{a_i r}\bar{T}^{H}_{ala},\\
\mathbf{R}_{a_i r} &= \text{orientation of robot arm at position } i,\\
\bar{T}^{S}_{p_i p_j r} &= \text{delta vector between } i \text{ and } j \text{ based on stereo triangulation,}\\
&= \mathbf{R}_{sr}\left(\bar{T}_{sp_j s} - \bar{T}_{sp_i s}\right),\\
\mathbf{R}_{sr} &= \text{orientation of PMJ w.r.t. the robot as per Equation 5.4,}\\
\bar{T}_{sp_i s} &= \mathfrak{f}^{\text{int}}\left(\bar{T}_{M_1 C_a M_1}, \bar{U}_{C_a p_i M_1}, \bar{T}_{M_1 C_b M_1}, \bar{U}_{C_b p_i M_1}\right),\\
\bar{T}_{M_1 C_n M_1} &= \text{camera position at mount 1 in PMJ CF,}\\
&= \bar{T}_{M_1 M_n M_1} + \mathbf{R}_{M_n M_1}\bar{T}_{MCM},\\
\bar{U}_{C_n p_i M_1} &= \text{UV from camera at mount } n \text{ to point } i \text{ in PMJ CF,}\\
&= \mathbf{R}_{M_a M_1}\mathbf{R}_{CM}\left(\mathfrak{f}^{\text{img}\to\text{vec}}(\mathfrak{f}^{\text{Brown}}(\bar{P}^{d}_{i,n}, \bar{DU}_{\text{params},n}), \bar{I}_{\text{params},n})\right),\\
\mathfrak{f}^{\text{int}} &= \text{as per Equation 3.20,}\\
\bar{DU}_{\text{params},m} &= \text{camera } m\text{'s DU parameters,}\\
\bar{I}_{\text{params},m} &= \text{camera } m\text{'s intrinsic parameters,}\\
\mathfrak{f}^{\text{img}\to\text{vec}} &= \text{as per Equation 3.17,}\\
\mathfrak{f}^{\text{Brown}} &= \text{as per Equation 3.25 (Section 3.4.6 for parameter details),}\\
\bar{P}^{d}_{i,j} &= \text{the LED centre (Section 3.4.5) of point } i \text{ by camera } j,\\
\mathbf{R}_{M_i M_1}, \bar{T}_{M_1 M_i M_1} &= \text{pose of the PMJ's mount } i \text{ (Section 5.4.1) w.r.t. PMJ,}\\
\mathbf{R}_{CM}, \bar{T}_{MCM} &= \text{pose of camera w.r.t. its mounting interface (Section 5.5.5),}\\
a &= \text{the number of the first stereo mount position, and}\\
b &= \text{the number of the second stereo mount position.}
\end{aligned}
$$

## 5.5 Camera calibration

This section describes the algorithms used to calibrate a camera once the APCCS has been calibrated. These algorithms then constitute the primary uses for which an operational APCCS will be used. All the intrinsic and extrinsic parameters will be measured with the exception of the pixel dimensions and their skewness. The dimensions are assumed known from the data sheet, and the pixel skewness is considered negligible with modern semiconductor foundry manufacturing techniques.

Section 5.5.1 describes the DU characterisation required for calculation of true UVs from camera imagery. Section 5.5.2 describes the reverse mapping (UD characterisation) which is required for real-time photogrammetric stitching. Section 5.5.3 describes how the focal length (matched to the determined distortion corrections) is found. Section 5.5.4 describes how the pose of mount w.r.t. the reference CF is determined by measuring the pose of the camera w.r.t. the reference and subtracting the camera w.r.t. the pose from it. Section 5.5.5 determines how this pose of the camera w.r.t. the mount is calculated. SEP in this manner allows the subsequent replacement of cameras in a system without repeating the calibration of the mount poses, which maybe beneficial from both a cost and time saving point of view.

### 5.5.1 Lens distortion correction

For lens distortion characterisation the maxim that straight lines in the real world must project onto straight lines in the image space (after distortion has been corrected) is used. To do this the robot (and its attached LED) is moved in a series of straight lines, stopping at several points along each line for an image to be captured. This results in $N$ lines being captured each of which has $M_i, \quad i \in (0, N-1)$ points. These points are referred to as $P^d_{i,j}$ indicating the original raw (i.e. distorted) image position of the $j$th point of the $i$th line.

Thereafter an arbitrary number of parameters for the Brown lens distortion

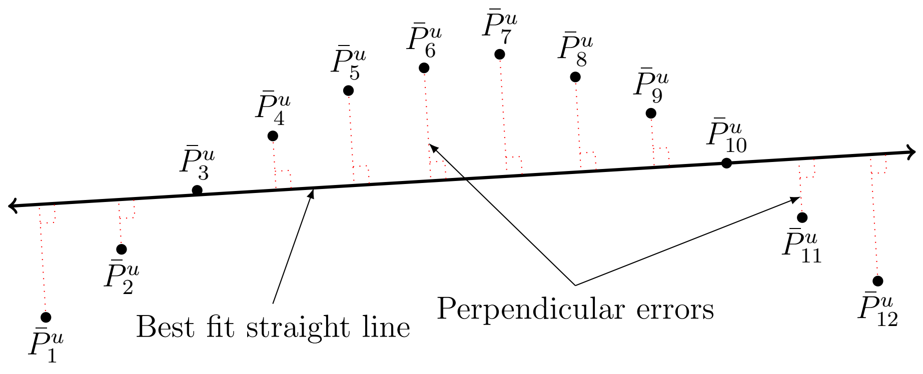


Figure 5.7: Lens distortion characterisation cost function.

model (Equation 3.25) can be numerically determined. Any numerical optimisation routine can be used, although some (such as Levenberg-Marquardt [23, 24]) perform worse due to their mathematical mechanics, the highly correlated nature of the parameters, and the residual noise inherent in the measurement of the input data.

A cost function is used to measure how straight a set of lines are. The cost function determines the best fit straight line through the point of each captured straight line using Equation 3.22. Thereafter the sum of the RMS distance of the points from their straight lines is calculated. This cost function will only ever be zero if all the points lie on straight lines, which is the aim of plumb-line based distortion modelling. This cost function was shown by de Villiers [15] to yield better results than alternatives such as minimising the coefficient of the second order term of the best fit quadratic through each straight line's points. This cost function is illustrated in Figure 5.7 and given mathematically in Equation 5.6:

$$\mathrm{C}^{D\to U}(\bar{DU}^s_{\text{params}}) = \sqrt{\frac{1}{\sum_{n=0}^{n<N_L} M_n} \sum_{n=0}^{n<N_L} \sum_{m=0}^{m<M_n} \left( (\bar{P}^u_{n,m} - [0, c_n]^T) \cdot \bar{d}_n \right)^2} \tag{5.6}$$

where

$\mathrm{C}^{D\to U}$ = the DU characterisation cost function,

$\bar{DU}^s_{\text{params}}$ = sensitivity scaled distortion correction parameters,

$$\begin{aligned}
N_L &= \text{the number of straight lines in the data captured,} \\
M_n &= \text{the number of points along the } n\text{th line,} \\
\bar{d_n} &= \text{unit direction vector orthogonal to the RMS line } n, \\
&= \frac{\left[1, \frac{-1}{m_n}\right]^T}{\| \left[1, \frac{-1}{m_n}\right] \|}, \\
m_n, c_n &= \text{coefficient of best fit line } n \text{ as per Equation 3.22,} \\
\bar{P}^u_{n,m} &= \text{the } m^{\text{th}} \text{ undistorted point on the } n^{\text{th}} \text{ straight line,} \\
&= \mathfrak{f}^{\text{Brown}}\left(\bar{P}^d_{n,m}, \bar{DU}^u_{\text{params}}\right) \text{ as per Equation 3.25,} \\
\bar{P}^d_{n,m} &= \text{the } m^{\text{th}} \text{ distorted point on the } n^{\text{th}} \text{ straight line,} \\
\bar{DU}^u_{\text{params}} &= \text{column vector of unscaled DU parameters,} \\
&= \bar{DU}^s_{\text{params}} \left(\bar{\kappa}_{\text{params}}\right)^T \text{, and} \\
\bar{\kappa}_{\text{params}} &= \text{column vector of scaling parameters as per Equation 5.7.}
\end{aligned}$$

As indicated in Equation 5.6, scaling the parameters to normalise the sensitivity of the gradient vector is required. These scaling parameters are in the pixel domain and so vary with camera resolution and field of view. In general each radial term's scaling factor is several orders of magnitude smaller than the previous radial term's scaling factor. The same is true for the infinite series portion of the tangential parameters. The first two radial parameters' scaling values are the same order of magnitude as the first radial scaling parameters and the principal point scaling factors are inversely proportional to the resolution of the image.

Equation 5.7 defines the scaling parameters vector:

$$\bar{\kappa}_{\text{params}} = \begin{bmatrix} \kappa_{hc} \\ \kappa_{vc} \\ \kappa_{R1} \\ \vdots \\ \kappa_{RN_R} \\ \kappa_{T1} \\ \vdots \\ \kappa_{TN_T} \end{bmatrix} \tag{5.7}$$

where

$$\begin{aligned} (\kappa_{hc}, \kappa_{vc}) &= \text{distortion centre scaling parameters,} \\ \kappa_{Ki} &= \text{scale factor for } i\text{th radial parameter,} \\ \kappa_{Pi} &= \text{scale factor for } i\text{th tangential parameter,} \\ N_R &= \text{the number of radial parameters, and} \\ N_T &= \text{the number of tangential parameters.} \end{aligned}$$

A starting vector is required for local optimisation (i.e. $\bar{V}^s$). Three common ways of determining this starting vector are:

1. Set all the parameters to 0.
2. Use prior knowledge to select approximate starting values.
3. Specify a range for each parameter and perform a coarse global optimisation, e.g. brute force or a genetic algorithm.

After initialisation the vector must be scaled so that the gradient is equally sensitive to a constant size perturbation in each dimension. This allows for a more accurate gradient estimation by the local optimisation procedure resulting in better characterisations. With reference to Equation 5.7, Equation 5.8 shows the scaling procedure:

$$\bar{DU}_i^{s*} = \bar{DU}_i^s / \bar{\kappa}_i \qquad \forall i \in [0, N] \cap \mathbb{Z} \tag{5.8}$$

where

$\bar{DU}^{s*}$ = the scaled starting parameter vector ready for numeric refinement,

$\bar{DU}^{s}$ = the chosen initial starting parameter vector,

$\bar{\kappa}$ = the desensitising scale vector as per Equation 5.7, and

$N$ = the number of parameters being refined.

### 5.5.2 Inverse distortion modelling

This section describes how the UD characterisation is performed. The characterisation is necessary to determine the image coordinate corresponding to a point in space. An example of where this is required is each pixel in a photogrammetrically stitched image (see Chapter 6 and de Villiers [12, 18]).

The method used in this work is that described by de Villiers et al. [16] whereby the matched sets of distorted and undistorted points produced during the DU characterisation are used to fit the parameters of a Brown distortion model [34] in the UD direction. This is performed by creating a cost function that measures the RMS error (in pixels) between the original distorted point (as captured during the robot movement sequence) and the redistorted point calculated by applying the current UD parameters to the undistorted point obtained during DU calibration (Section 5.5.1). Such a cost function will only ever achieve its minimum of zero, if all the undistorted-and-then-redistorted points exactly coincide with the original distorted points. This cost function is expressed mathematically as:

$$\mathrm{C}^{U\to D}(\bar{UD}^{s}_{\text{params}}) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\|\bar{P}^{d}_{i} - \bar{P}^{d*}_{i}\|} \tag{5.9}$$

where

$\mathrm{C}^{U\to D}$ = the UD characterisation cost function,

$N$ = the number of distorted points,

$\bar{P}^{d}_{i}$ = the $i^{\text{th}}$ distorted image point,

$\bar{P}^{d*}_{i}$ = the $i^{\text{th}}$ undistorted and then redistorted image point,

$$
\begin{aligned}
&= \mathfrak{f}^{\text{Brown}}\left(\mathfrak{f}^{\text{Brown}}\left(\bar{P}_i^d, \bar{DU}_{\text{params}}\right), \bar{UD}^s_{\text{params}}\left(\bar{\kappa}_{\text{params}}\right)^T\right) \\
\mathfrak{f}^{\text{Brown}} &= \text{as per Equation 3.25 (Section 3.4.6 for parameter details),} \\
\bar{UD}^s_{\text{params}} &= \text{the scaled reverse distortion correction parameters,} \\
\bar{\kappa}_{\text{params}} &= \text{vector of scaling parameters as per Equation 5.7, and} \\
\bar{DU}_{\text{params}} &= \text{the camera's DU parameters as per Section 5.5.1.}
\end{aligned}
$$

The starting vector is given as either all zeroes or the negative of the DU parameters, with the exception of the distortion centre where the DU centre is used. This starting point is then fed into a local optimisation algorithm. The Leapfrog algorithm (see Section 3.2.2.4) is used due to its noise robustness and ability to find superior local minima.

### 5.5.3 Focal length determination

The focal length determination makes use of the tetrahedron problem detailed in Section 3.4.7 and summarised in Equation 3.29. Note that Equation 3.29 makes use of Equation 3.25 (which is dependent on the distortion parameters of the lens which are assumed to have already been calibrated via the methods described in Section 5.5.1). Equation 3.29 also makes use of Equation 3.17 which makes use of the focal length which is the subject of this characterisation.

The robot arm is used to place the LED in the FOV of the camera such that a number of tetrahedrons are created. In this research 20 tetrahedrons were created in four groups of five. Each group had the tetrahedrons centred at a different location, with the groups' central positions forming a '+'. At each location the tetrahedrons were angularly offset such that the camera's optical axis was not normal to the base of the tetrahedron. The tetrahedrons in a group were respectively angled to look up-to-the-right, down-to-the-right, down-to-the-left, and up-to-the-left as seen from the camera's point of view.

To determine the focal length the camera was placed in two positions to view the tetrahedrons. The camera was rigidly placed in the first position and then viewed all the tetrahedrons, after which it was placed in the second position and it again viewed the robot moving through the exact same tetrahedron sequence. The camera was stationary at each position while viewing the tetrahedrons. This means that the relative displacement of the camera was constant. Since the robot arm was used to move the energy source to each subsequent position for each tetrahedron, the translations of the tetrahedron points are known. If the distortion parameters are already known, then the focal length remains the only parameter required by Equation 5.10 via its dependence on Equation 3.17.

The orientation of the robot end effector is kept constant, while it is being moved to each of the tetrahedrons' vertices. This is to allow the focal length to be determined even if the LED translation offset from the end effector is unknown. This in turn allows a camera to be used to determine the LED spatial offset (Section 5.4.2) once the focal length has been determined.

For an assumed focal length the position and orientation of the camera relative to the robot reference (which is the CF in which the LED's translations, i.e. the $\bar{T}_{icc}$'s of Equation 3.29, are known) can be calculated. At the correct focal length, the locus of calculated camera positions will be the smallest possible, theoretically zero if there are no errors in the LED image position measurement and DU characterisation. This can already be used as a cost function to determine the ideal focal length. This sensitivity to focal length is increased by comparing the relative position and orientation of the camera at the second position to the camera at the first position for each tetrahedron. This is possible due to the high repeatability [72] of the robot arm used (an ABB IRB120). For each tetrahedron $i$, the pose of the cameras at each of the two mounts (mount $a$ and mount $b$) is calculated. A measure of the similarity between the two poses for that tetrahedron ($\Delta_i$) is then calculated. This measure is not expected to be zero (as the camera is on different physical mounts) but it should ideally be constant over all the tetrahedrons. The variation in the similarity measure is the cost function that is minimised. This measure will reach its minimum of zero, when the relative pose of the two camera is constant, which will only happen if the DU characterisation is accurate and the focal length is correct. Equation 5.10

provides the mathematics:

$$\mathrm{C}^F(FLen) = \sqrt{\frac{1}{N_{tet}}\sum_{i=1}^{N_{tet}}[\Delta_i^2] - \left(\frac{1}{N_{tet}}\sum_{i=1}^{N_{tet}}[\Delta_i]\right)^2} \tag{5.10}$$

$$\begin{aligned}
&\text{where}\\
\mathrm{C}^F &= \text{the cost function to determine the focal length,}\\
N_{tet} &= \text{the number of tetrahedrons observed,}\\
\Delta_i &= \text{pose dissimilarity of } a \text{ and } b \text{ for tetrahedron } i,\\
&= k_1\theta_i + k_2\|\bar{T}_{c_{a,i}c_{b,i}r}\|,\\
\theta_i &= \mathfrak{f}^{\mathrm{RtoAA}}(\mathbf{R}^T_{c_{a,i}r}\mathbf{R}_{c_{b,i}r}),\\
\bar{T}_{c_{a,i}c_{b,i}r} &= \bar{T}_r c_{b,i} r - \bar{T}_r c_{a,i} r,\\
(\bar{T}_r c_{a,i} r, \mathbf{R}_{c_{a,i}r}) &= \mathfrak{f}^{tet}\left(\bar{U}_{c_{a,i}jc_{a,i}} \forall j \in [1,4], \bar{T}_{t_{i}jr} \forall j \in [1,4]\right),\\
\mathfrak{f}^{tet} &= \text{as per Equation 3.29,}\\
(\bar{T}_r c_{b,i} r, \mathbf{R}_{c_{b,i}r}) &= \mathfrak{f}^{tet}\left(\bar{U}_{c_{b,i}jc_{b,i}} \forall j \in [1,4], \bar{T}_{t_{i}jr} \forall j \in [1,4]\right),\\
\bar{U}_{c_{a,j}ic_{a,j}} &= \mathfrak{f}^{\mathrm{img}\to\mathrm{vec}}(\mathfrak{f}^{\mathrm{Brown}}(\bar{P}^d_{a,i,j}, \bar{DU}_{\mathrm{params}}), \bar{I}_{\mathrm{params}}),\\
\bar{U}_{c_{b,j}ic_{b,j}} &= \mathfrak{f}^{\mathrm{img}\to\mathrm{vec}}(\mathfrak{f}^{\mathrm{Brown}}(\bar{P}^d_{b,i,j}, \bar{DU}_{\mathrm{params}}), \bar{I}_{\mathrm{params}}),\\
i &\in [1, N_{tet}] \text{ and denotes the tetrahedron number,}\\
j &\in [1,4] \text{ and denotes tetrahedron vertex number,}\\
\mathfrak{f}^{\mathrm{img}\to\mathrm{vec}} &= \text{as per Equation 3.17 which is dependent on the focal length,}\\
\mathfrak{f}^{\mathrm{Brown}} &= \text{as per Equation 3.25,}\\
\bar{P}^d_{x,i,j} &= \text{the LED image coordinates of tetrahedron } i \text{ vertex } j\\
&\qquad \text{seen from camera position } x \text{ (Section 3.4.5),}\\
\bar{U}_{c_{x,j}ic_{x,j}} &= \text{camera's UV from position } x \text{ to vertex } i^{\text{th}} \text{ of tetrahedron } j,\\
\bar{DU}_{\mathrm{params}} &= \text{the distortion parameters (Section 3.4.6), and}\\
\bar{I}_{\mathrm{params}} &= \text{the intrinsic parameters, including the focal length}\\
&\qquad \text{(Section 3.4.2).}
\end{aligned}$$

Weighting values for Equation 5.10 of $K_1 = 10$ and $K_2 = 1.0$ were used. In order to find the ideal focal length, a range centred around the lens's claimed/designed focal length needs to be searched for the minimum value of Equation 5.10. This can take the form of any line search technique such as Powell's method (Figure 3.6) or the golden ratio search (Equation 3.1). Equation 5.10 is dependent on the tetrahedron pose determination (Equation 3.29) which selects the best of up to four possible real solutions (Equation 3.28). Equation 3.28 can potentially choose a different solution of Equation 3.27 for small changes in the focal length. This causes the cost function to be non-continuous, but still one dimensional. Therefore a coarse-to-fine brute-force search was used.

No previous focal length calibrations based on the clustering of analytically determined camera poses using hypothesised focal lengths were found in literature. This method is thus considered to be a novel approach and successfully reduces the dimensionality of the search space when the camera pose is determined in Section 5.5.4.

### 5.5.4 Mount pose determination

Determining the pose of the mount relative to the reference CF requires knowledge of the camera pose w.r.t. its mount and the global pose of the camera in the reference CF. The determination of the camera pose offset is described in Section 5.5.5. This section therefore concentrates on the determination of the global camera pose.

The DU parameters and focal length are required to determine this global camera pose. The focal length can either be determined as per Section 5.5.3 or added as the seventh parameter being numerically refined (it affects the $\bar{U}^1_{cic}$ vectors). The spatial offset of the LED relative to the end effector is also required (the robot reports the pose of the end effector, not of the attached LED). The knowledge of the LED spatial offset is not required if the the orientation of the arm is kept constant and the resultant constant spatial error (which is equal to the LED spatial offset) is acceptable. The constant unknown spatial offset is irrelevant when calibrating a stereo pair, as the determination of the relative positions of the mounts will be correct since the two spatial errors in the mount pose relative to the robot will cancel.

The global position is determined by minimising the dissimilarity between two bundles of vectors created by the camera observing the robot arm through a sequence of positions. The first bundle, termed $\bar{U}^1_{cic}$, is based on the intrinsic parameters and the captured LED centres during a robot movement sequence. The second bundle, called $\bar{U}^2_{cic}$, is created by hypothesising a pose of the camera relative to the robot arm and calculating the vector from this hypothesised pose to each LED using the reported pose of the robot and (possibly) the LED spatial offset. What is then desired is to maximise the similarity of the directions of the corresponding vectors of the two bundles. This will result in finding the camera pose that best explains the positions of the LEDs in the camera's image. The cost function used is the sum of the angles between all the corresponding vectors in the two bundles. These angles are calculated from the inverse cosine of dot product of the normalised corresponding vectors. The negative of the dot product itself could be used as a cost function too, but is insensitive to small angular differences. Equation 5.11 expresses the implemented cost function mathematically:

$$\mathrm{C}^{EP}(\mathbf{R}^H_{cr}, \bar{T}^H_{rcr}) = \sum_{i=0}^{n-1} \left( \cos^{-1}(\bar{U}^1_{cic} \bullet \bar{U}^2_{cic}) \right) \tag{5.11}$$

where

$$\begin{aligned}
\mathrm{C}^{EP} &= \text{the extrinsic parameter cost function,}\\
\mathbf{R}^H_{cr} &= \text{hypothesised orientation of the camera w.r.t. robot arm,}\\
\bar{T}^H_{crc} &= \text{hypothesised position of robot arm w.r.t. camera,}\\
\bar{U}^1_{cic} &= \text{image processing based UVs,}\\
&= \mathfrak{f}^{\text{img}\to\text{vec}}(\mathfrak{f}^{\text{Brown}}(\bar{P}^d_i, \bar{DU}_{\text{params}}), \bar{I}_{\text{params}}),\\
\mathfrak{f}^{\text{img}\to\text{vec}} &= \text{as per Equation 3.17,}\\
\mathfrak{f}^{\text{Brown}} &= \text{as per Equation 3.25,}\\
\bar{P}^d_i &= \text{distorted LED centre (Section 3.4.5) at position } i,\\
\bar{DU}_{\text{params}} &= \text{the camera DU parameters (Section 3.4.6),}\\
\bar{I}_{\text{params}} &= \text{the camera intrinsic parameters (Section 3.4.2),}
\end{aligned}$$

$$\begin{aligned}
\bar{U}^2_{cic} &= \text{UVs based on hypothesised camera pose,} \\
&= \bar{T}_{cic}/|\bar{T}_{cic}|, \\
\bar{T}_{cic} &= (\mathbf{R}^H_{cr})^T \left(\bar{T}_{rl_ir} - \bar{T}_{rcr}\right), \\
\bar{T}_{rl_ir} &= \text{the LED position w.r.t. the robot CF,} \\
&= \mathbf{R}_{a_ir}\bar{T}_{ala} + \bar{T}_{ra_ir}, \\
\mathbf{R}_{a_ir} &= \text{orientation of the robot arm at pose } i, \\
\bar{T}_{ala} &= \text{position of LED w.r.t. end effector, and} \\
\bar{T}_{ra_ir} &= \text{the position of the robot arm at pose } i.
\end{aligned}$$

This cost function is then numerically optimised using a robust algorithm such as Fletcher-Reeves [21] or (as in this work) Leapfrog [26]. This cost function is slower to evaluate but more accurate than merely using the negative of the dot product, due to the increased sensitivity to almost-parallel vectors provided by the (computationally intensive) inverse cosine function. It should be noted that if the focal length is being determined in conjunction with camera pose, and all the LED positions lie in a single plane, then Equation 5.11 becomes ill-conditioned as the camera angle relative to this plane approaches 90°. When the camera is perfectly orthogonal the focal length and displacement along the camera axis cannot be uniquely determined, only their ratio can be found.

Once the global camera position has been calculated, one can simply determine the pose of the mount w.r.t. the robot using the camera w.r.t mount offset:

$$\begin{aligned}
\mathbf{R}_{mr} &= \mathbf{R}_{cr}\mathbf{R}^T_{cm}, \text{ and} \\
\bar{T}_{rmr} &= \bar{T}_{rcr} - \mathbf{R}_{mr}\bar{T}_{mcm} \qquad (5.12)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{R}_{mr}, \bar{T}_{rmr} &= \text{the desired pose of the mount,} \\
\mathbf{R}_{cr}, \bar{T}_{rcr} &= \text{camera pose w.r.t. reference CF as per Equation 5.11, and} \\
\mathbf{R}_{mt}, \bar{T}_{rmr} &= \text{camera pose w.r.t. mount as per Equation 5.13.}
\end{aligned}$$

### 5.5.5 Camera mount pose offset determination

This section describes how the pose of the camera w.r.t. the camera's repeatable mounting bracket is determined. In terms of optics nomenclature the determined pose is the orientation of the optical axis (with roll defined by the CCD) with the translation of the entrance pupil all expressed w.r.t. the CF defined by the mechanical mounting interface of the camera. This calibration requires the outputs of the DU and focal length characterisations. In this calibration it is explicitly assumed that orientation of the end effector is kept constant, so that the LED spatial offset may be ignored. This allows the results of this calibration to be used to subsequently calculate the LED offset (Section 5.4.2). The PMJ is used for this calibration. The PMJ's pose w.r.t. the robot CF is not required to be known a priori.

This calibration requires the camera to be placed on each of the mounting brackets of the PMJ (whose relative poses are all known a priori) and to observe the same sequence of LED positions. For each possible unique pair of mounts on the PMJ, the position of each LED (w.r.t. the PMJ CF) in the movement sequence is triangulated using stereo vision. The average of all these triangulated positions is compared to the provided position of the end effector for the corresponding position in the arm movement sequence. If the average triangulated LED positions exactly coincide with the positions reported by the robot then the poses of the cameras on the mount must be correctly specified in the robot's CF.

The stereo triangulation requires that the captured LED pixel position be converted into a 3D vector. This pixel to vector conversion is why the camera DU characterisation and focal length are required. However these vectors are calculated in the local camera CF, and the orientation of the camera on each mount w.r.t. the robot is required. This camera to robot pose requires knowledge of the PMJ pose w.r.t. the robot CF, knowledge of each mount's pose w.r.t. the PMJ (known from prior calibration: Section 5.4.1), and knowledge of the camera pose w.r.t. the bracket. In addition to knowledge of the direction vectors, triangulation requires knowledge of a point along the vectors, which in this case corresponds to the camera position. This camera position is the summation of the PMJ, bracket and camera spatial offsets.

The cost function is thus dependent on both the pose of the PMJ w.r.t. the robot and the pose of the camera w.r.t the mount. The cost function is the RMS error (over all the points in the robots movement sequence) of the difference between the average triangulated LED position and the reported end effector position (the end effector pose is kept constant). Equation 5.13 provides the mathematics:

$$\mathrm{C}^{MP}\left(\mathbf{R}^{H}_{jr}, \bar{T}^{H}_{rjr}, \mathbf{R}^{H}_{cm}, \bar{T}^{H}_{mcm}\right) = \sqrt{\frac{1}{N_p - 1}\left(\sum_{i=0}^{N_p-1}\left\|\bar{T}_{rp_ir} - \frac{2}{N_m^2 - N_m}\sum_{j=0}^{N_m-2}\sum_{k=j+1}^{N_m-1}\bar{T}^{S}_{rp_{i,j,k}r}\right\|\right)} \tag{5.13}$$

where

$\mathrm{C}^{MP}$ = the mount pose cost function,

$\mathbf{R}^{H}_{jr}, \bar{T}^{H}_{rjr}$ = hypothesised pose of PMJ w.r.t. robot CF,

$\mathbf{R}^{H}_{cm}, \bar{T}^{H}_{mcm}$ = hypothesised pose of camera w.r.t. mount expressed in the mount's CF,

$N_p$ = number of LED image coordinates found,

$N_m$ = number of mounting points on the PMJ (4 in this work),

$\mathbf{R}_{p_ir} = \mathbf{R}_{p_jr} \forall i, j \in [0, N_p - 1]$,

$\bar{T}_{rp_ir}$ = position of end effector as reported by the robot arm,

$\bar{T}^{S}_{rp_{i,j,k}r}$ = stereo triangulation of LED at position $i$ as calculated using cameras at mounts $j$ and $k$,

$= \mathfrak{f}^{\text{int}}\left(\bar{T}_{rc_jr}, \bar{U}_{c_jp_ir}, \bar{T}_{rc_kr}, \bar{U}_{c_kp_ir}\right)$,

$\bar{T}_{rc_lr}$ = position of camera at mount $l$ w.r.t. robot CF,

$= \bar{T}^{H}_{rjr} + \mathbf{R}^{H}_{jr}\left(\bar{T}_{jm_lj} + \mathbf{R}_{m_lj}\bar{T}^{H}_{mcm}\right)$,

$\bar{U}_{c_lp_ir}$ = UV from camera at mount $l$ to LED $i$ in robot's CF,

$= \mathbf{R}^{H}_{jr}\mathbf{R}_{m_lj}\mathbf{R}^{H}_{cm}\left(\mathfrak{f}^{\text{img}\rightarrow\text{vec}}\left(\mathfrak{f}^{\text{Brown}}\left(\bar{I}^{d}_{i,l}, \bar{DU}_{\text{params},l}\right), \bar{I}_{\text{params},l}\right)\right)$,

$\mathfrak{f}^{\text{int}}$ = as per Equation 3.20,

$\mathfrak{f}^{\text{img}\rightarrow\text{vec}}$ = as per Equation 3.17,

$$\begin{aligned}
\mathfrak{f}^{\text{Brown}} &= \text{as per Equation 3.25,}\\
\bar{DU}_{\text{params},l} &= \text{DU corrections parameters for camera } l \text{ (Section 3.4.6),}\\
\bar{I}_{\text{params},l} &= \text{intrinsic camera parameters for camera } l \text{ (Section 3.4.2),}\\
\bar{I}^{d}_{i,l} &= \text{LED pixel centre } i \text{ (Section 3.4.5) as seen from mount } l\text{, and}\\
\mathbf{R}_{m_l j}, \bar{T}_{j m_l j} &= \text{known pose of mount } l \text{ w.r.t. PMJ (Section 5.4.1).}
\end{aligned}$$

The cost function of Equation 5.13 is seeded with an initial 12 parameters which are then refined using a local optimisation algorithm (in this research Leapfrog [26] is used). The initial parameters can be determined with coarse estimates obtained from a tape measure for the spatial parameters and rough estimates for the angular parameters. No previous work on separating the extrinsic parameters into two poses based on only the relative, rather than absolute, poses of several camera mounts could be found in literature. This calibration is thus considered novel.

An alternative cost function based on first determining the pose of each camera w.r.t. the robot using Equation 5.11 was also considered. In this scenario, the hypothesised PMJ w.r.t. the robot pose, known mount w.r.t. the PMJ pose, and hypothesised camera w.r.t mount pose are used to calculate hypothesised camera w.r.t the robot poses. The two hypothesised poses are then numerically refined until the resultant camera poses are maximally similar to those provided by Equation 5.11. This was experimentally seen to provide worse results than the cost function detailed above, possibly due to the lack of rigorous methods of determining the similarity of two poses.

# Chapter 6

# Application of photogrammetric calibration parameters

This chapter aims to address the second research question proposed in Section 1.2: *“Are the calibration parameters produced by such a system suitable for real world applications?”* This chapter provides two examples of real-world photogrammetric applications. Cameras are calibrated with the APCCS for each application and the resultant accuracy of the application is used as a yardstick to determine if the APCCS is indeed a useful practical system.

The primary example, photogrammetric stitching, is presented in Section 6.1. This section shows how to both register and then blend images from cameras with different resolutions, FOVs and spectrums to create a seamless false colour panorama. Two different systems with different numbers of cameras arranged in different physical configurations are evaluated. The accuracy of the stitching is assessed both subjectively via the resultant panoramas and objectively via analysis of the accuracy which image features visible to multiple cameras are overlaid on the panorama.

The second example application, optical helmet tracking is presented with less rigour in Section 6.2. The reason for the decreased rigour is due to the results already being published [7], the sensitivity of such military related equipment, and to decrease the size and scope of this thesis. The section describes the physical apparatus of a multi-camera Helmet Tracker System (HTS), the basic mathematical principles of a HTS and the results obtained for a simple laboratory system which was constructed for this test.

## 6.1 Photogrammetric stitching

Stitching is the creation of a panoramic image by assembling smaller images into a larger image. All forms of stitching, which is also referred to as mosaicking, consist of a registration step followed by a blending step. Registration determines which pixels from the input images correspond to each pixel in the output stitch (Section 6.1.1). Blending determines the relative weightings of each camera for each pixel of the stitch. This is to equalise the variance in brightness and colour balance between adjacent cameras, to create false colour images for multispectral camera arrays, and to minimise any errors in registration. Section 6.1.2 discusses blending in more detail. The quantitative measure of how accurately the stitch was performed is given in Section 6.1.3. Section 6.1.4 provides both example panoramas as subjective proof of the fidelity of the stitching algorithm as well as the quantitative accuracy. Section 6.1.5 summarises the stitching work and places its results in context.

Table 6.1 shows the number of parameters required for a typical multi-camera staring array surveillance system. These parameters exclude blending parameters and parameters not measured, such as the pixel dimensions which are obtained from the data sheet.

### 6.1.1 Photogrammetric registration

The registration step of stitching entails determining the correct relative alignment, rotation, scaling and shear of the input images. Wide FOV cameras also need to have their distortion corrected or the inward bending of objects will prevent proper registration. Figure 6.1 shows an example where the door frame in

Table 6.1: Photogrammetric parameters required for stitching.

| **Item** | **Calibration** | **Number of parameters** | **Note** |
|---|---|---|---|
| Camera | Total | 33 | - |
| | DU calibration | 10 | 5 radial, 3 tangential, distortion centre |
| | UD calibration | 10 | 5 radial, 3 tangential, distortion centre |
| | Focal length | 1 | - |
| | Camera to mount pose | 6 | 3 angular, 3 spatial |
| | Mount to reference pose | 6 | 3 angular, 3 spatial |
| Stitching | Geometry | 5 | Radius, plane normal and plane distance (as per Equation 6.1) |
| | Resolution | 6 | Azimuth minimum, maximum and step size. Elevation minimum, maximum and step size. |
| Complete system | 4 cameras | 143 | - |
| | 8 cameras | 275 | - |

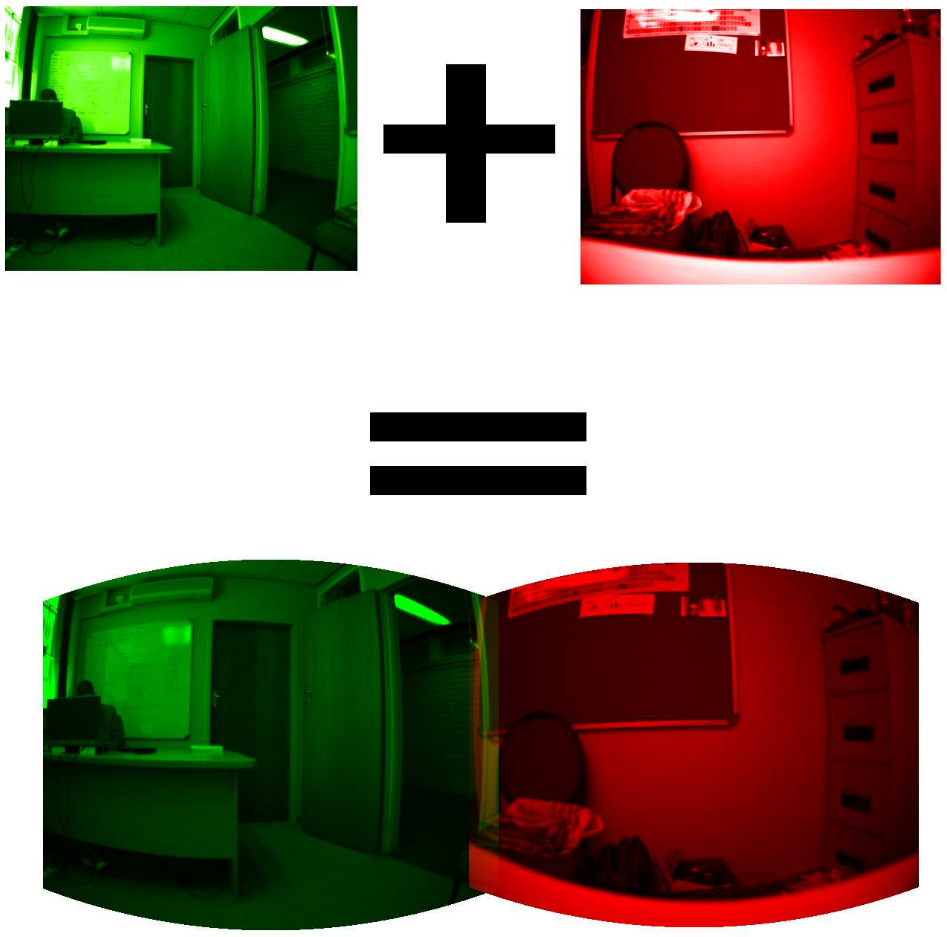


Figure 6.1: Distortion induced stitching errors.

the overlap cannot be correctly aligned with the raw distorted images. The final output of registration is the ability to determine which pixels from which images, correspond to which pixel in the panorama (and to each other).

It is possible to stitch images together from non-calibrated cameras. This is called feature based stitching and uses the content of each image to determine the registration. Image features are found in each image. These are image points that are repeatably and accurately locatable, as well as uniquely identifiable despite slight changes in magnification, rotation, illumination and perspective. Popular image feature algorithms include the seminal SIFT [81] and

SURF [82]. These are floating point algorithms and tend to be slow and difficult to parallelise. More recently simpler binary algorithms that are less robust but more parallelisable have been developed, these include Binary Robust Invariant Scalable Keypoints (BRISK) [83] and Binary Robust Independent Elementary Features (BRIEF) [84]. Recent work has managed to combine much of the robustness of the floating point algorithms with the speed of the binary algorithms, a preeminent example being Binary Features from Robust Orientation Segment Tests (BFROST) [85].

Once the chosen features have been found, corresponding matching features in the overlapping portion of each adjacent camera pair's FOV are determined. Thereafter, the transformations are applied to each image so that the matching feature pairs coincide in the stitch image. This allows the relative camera motion, changes to the FOV/magnification to be taken into account, and a single camera to produce a panorama by changing its viewing direction over time.

Despite the advantage listed above, there are several disadvantages to feature based stitching. Chief among them is that finding image features, matching them across images and then determining the mapping between images to create the panorama are all slow and typically nondeterministic processes. Furthermore, low contrast or poorly exposed images may not have enough detail to find sufficient usable features with which to do the registration. Feature based stitching also requires large overlaps of the camera FOVs and thus more cameras per solid angle are needed. Finally, the stitch may not be created in a geometrically representative manner. This would then not allow the effective intrinsic parameters of the stitch to be determined.

Photogrammetric registration is the alternate to feature based stitching. By calibrating the camera's intrinsic and extrinsic parameters, the stitching process can be made quicker, deterministic and create a known geometry that allows the corresponding world vector of any stitch point to be easily determined. The trade-off is that the stitching becomes sensitive to relative camera motion.

Photogrammetric registration works by ray-tracing each point of a theoretical constellation of points onto each camera's image. Each point in the constellation is uniquely associated with (or generated from the coordinates of) a pixel in the stitched image. The translation of each point is calculated relative to each camera and then projected into the camera's virtual inverted image plane. This spatial coordinate is then converted into pixel space to yield an ideal undistorted pixel position for that camera. These undistorted pixel coordinates are converted into distorted coordinates using each camera's unique UD parameters. These distorted coordinates are then used to look up the colour value for that camera for that point in space.

For any physically realised system, the cameras will have a non-zero relative displacement or baseline. This means that the stitch is depth sensitive, particularly in the overlapping regions. Therefore, if the real world distance of an object is not at the exact distance at which the hypothetical stitch position for that pixel is calculated, then the incorrect pixel in the camera is used for that stitch point. This causes blur in the overlap regions whose magnitude is dependent on the difference between the object and the stitch distance. This can be minimised by attempting to alter the distance associated with each pixel to match the expected scene. A multi-geometry stitch surface based on the intersection of a plane and sphere was developed and presented at a conference [12] in 2013. The multi-geometry stitch better simulated typical scenarios of camera arrays in a terrestrial deployment and was shown to drastically improve the stitch fidelity.

In this research, the registration includes the biasing of each camera's weighted contribution such that for each pixel, the camera which has that point closest to its centre is favoured. This has two effects: firstly, it causes a smoother transition in the white balancing of adjacent cameras. Secondly, the weighting causes pixels in a camera's peripheral FOV, where distortion correction is typically the least accurate, to contribute less to the stitch.

To create a Mercator projection, each stitch pixel is assumed to subtend a constant angular width and height. Using the procedures described in the preceding paragraphs to vary the distance of the stitch surface results in the modified

Mercator projection used here. Equation 6.1 provides the mathematics:

$$\bar{I}^{s}_{(r,g,b,i)}\left(h_s, v_s\right) = \frac{\sum_{j=0}^{N_{cam}-1}\left[\bar{I}^{(c_j)}_{(r,g,b,i)}\left(\bar{P}^{(c_j)}_{(h,v)}\right)W^{c_j}\left(\bar{P}^{(c_i)}_{(h,v)}\right)\right]}{\delta + \sum_{j=0}^{N_{cam}-1}\left[W^{c_j}\left(\bar{P}^{(c_j)}_{(h,v)}\right)\right]} \tag{6.1}$$

where

$$\begin{aligned}
N_{cam} &= \text{the number of cameras in the stitch,}\\
\bar{P}^{(c_i)}_{(h,v)} &= \mathfrak{f}^{\text{Brown}}(\mathfrak{f}^{\text{P}\to\text{U}}(\mathbf{R}^{T}_{c_i r}\left(D^{*}\bar{U}_{rsr}(h_s, v_s) - \bar{T}_{rc_i r}\right), \bar{I}_{\text{params},i}), \bar{UD}_{\text{params},i}),\\
\mathfrak{f}^{\text{Brown}} &= \text{as per Equation 3.25 (Section 3.4.6 for parameter details),}\\
\mathfrak{f}^{\text{P}\to\text{U}} &= \text{as per Equation 3.18,}\\
\bar{UD}_{\text{params},j} &= \text{camera } j\text{'s UD parameters,}\\
\bar{I}_{params,j} &= \text{camera } j\text{'s intrinsic parameters,}\\
\bar{I}^{(c_j)}_{(r,g,b,i)}(h, v) &= \text{colour quadruplet of camera } j \text{ at coordinates } (h, v),\\
\bar{U}_{rsr}(h_s, v_s) &= \begin{bmatrix} \cos(El_{\min} + v_s \times El_{\Delta})\cos(Az_{\min} + h_s \times Az_{\Delta})\\ \cos(El_{\min} + v_s \times El_{\Delta})\sin(Az_{\min} + h_s \times Az_{\Delta})\\ \sin(El_{\min} + v_s \times El_{\Delta}) \end{bmatrix},\\
(Az_{\min}, El_{\min}) &= \text{aiming angles of top left stitch pixel,}\\
(Az_{\Delta}, El_{\Delta}) &= \text{horizontal and vertical angle each stitch pixel subtends,}\\
D^{*}(h_s, v_s) &= \min(D^{s}(h_s, v_s), D^{p}(h_s, v_s)),\\
D^{s}(h_s, v_s) &= \text{distance from origin to stitch sphere, i.e. the stitch radius,}\\
D^{p}(h_s, v_s) &= \text{distance from origin to stitch plane,}\\
&= \frac{D_{pr}}{\bar{U}_{rsr}(h_s, v_s) \bullet \bar{U}_{rpr}},\\
D_{pr} &= \text{distance of plane from stitch origin,}\\
\bar{U}_{rpr} &= \text{outward pointing normal UV of plane,}\\
W^{c_j}\left(\bar{P}^{(c_j)}_{(h,v)}\right) &= \text{weight of camera } j\text{'s contribution,}\\
&= \max\left(0.0, \bar{P}^{(c_i)}_{h}(R_h - \bar{P}^{(c_i)}_{h})\right) \times \max\left(0.0, \bar{P}^{(c_i)}_{v}(R_v - \bar{P}^{(c_i)}_{v})\right),\\
(R_h, R_v) &= \text{pixel resolution of camera } j\text{, and}\\
\delta &= \text{a tiny positive value to ensure a nonzero denominator.}
\end{aligned}$$

Equation 6.1 is a summary of the improved stitching algorithm [12], including quicker blending and multiple geometry stitching. For a more detailed discussion on each step of the stitch algorithm, implementation considerations, and timing results the reader is referred to the two papers [12, 18] published that detail the algorithm.

The next question that arises is that of choosing the stitching parameters. Some of the parameters are fairly straightforward: the azimuth and elevation range of the stitch should cover the combined FOVs of the cameras. The range may be decreased slightly to clip the bubble like bowing at the top of the panorama, which is due to projecting a rectangular FOV onto the inside of a sphere. The azimuth and elevation step sizes are chosen based on a combination of several criteria: the maximum allowable size of the stitch (constrained to 8192 pixels per side in current GPUs), the resolution of the screen on which the stitch is displayed, the effective angular resolution of the cameras used in the stitch, and the size of the smallest object that is required to be detectable in the stitch. The stitch radius is typically chosen according to the needs of the end application: for instance a ground based system is unlikely to have long view lines and may have multiple close objects, thus the stitch radius will be decreased. Conversely an airborne system may have a large stitch radius as it is unlikely to encounter any close objects. The stitching plane in most cases will coincide with the local terrain. On level ground or calm waters, the stitching plane's normal vector will point downwards. The planar distance will be set to match the height of the system above the ground. On a moving system, both the plane normal and distance could be updated by inertial systems. On non level terrain the plane normal and distance should be set to match the best approximation of the local terrain.

### 6.1.2 Blending different spectra

Blending is the second stitching step. It entails the creation of an output colour for each stitch pixel based on the corresponding input pixel/s in the set of input images. The correspondence of input image pixels to output stitch pixels is the result of the registration step (Section 6.1.1). The blending needs to cater for images with different exposures, white balances, and potentially even different sensitivity spectra.

In this work the colour balancing and centre-bias weighting of each camera's contribution in overlapping portions of the FOV is performed by the primary stitching algorithm (Equation 6.1). The output of the stitching algorithm is assumed to be a colour quadruplet consisting of red, green, blue and LWIR values for each pixel. The purpose of this section is the creation of false colour images to display these 4 colour images on a typical 3 colour display. This is referred to as image fusion.

Intuitively one is inclined to tint hotter objects red and colder objects blue. For greyscale visual imagery this works well. However, for colour visual imagery this results in ambiguities such as whether a red object in the fused image is red in colour and of ambient real world temperature, or whether it is white in colour and particularly hot in the real world. The solution to this problem seems to be application specific.

An evaluation was performed for a surveillance scenario. Three algorithms as well as the raw visual and raw IR panoramas were evaluated by a panel of 32 assessors. Four different scenarios were evaluated: three indoor scenes with controlled lighting conditions ranging from dark to brightly lit, and one sunny outdoor scene. The AHP [86] was used to determine the best algorithm for each scenario. This entails directly comparing every possible permutation of pairs of fused panoramas side by side. For each pair, each assessor independently selects which fusion algorithm is better and quantifies by how much the preferred algorithm is superior.

The full details of the comparison including the analysis methodology, mathematics and details of all the fusion algorithms were published and can be found in the paper [14]. This work only presents the results sufficient to show that the panoramas created by arrays of visual and thermal cameras are created successfully. This in turn shows that the APCCS can successfully and accurately calibrate cameras of different spectra. Results for well lit scenarios, where the

Table 6.2: Values of factors used in fusion algorithms.

| **Scene** | Algorithm | $\alpha_{H1}$ | $\alpha_{H2}$ | $\alpha_{C1}$ | $\alpha_{C2}$ | $\kappa_1$ | $\kappa_2$ | $\kappa_3$ |
|---|---|---|---|---|---|---|---|---|
| **Outside / well lit** | TTGS | 0.65 | 1.0 | 0.0 | 0.1 | | | |
| | TBHO | 0.65 | 1.0 | 0.0 | 0.1 | 0.3 | 0.9 | 0.5 |

Thermal Tinged Gray Scale (TTGS) and Thermal Based Hue Offset (TBHO) algorithms were found to provide the best results [14], are presented and explained below. Both of these fusion algorithms use several weighting values and parameters for hot and cold fuzzy membership functions. These values were empirically tuned and are given in Table 6.2.

### 6.1.2.1 Thermal tinged greyscale

The TTGS algorithm creates a false coloured image based on a greyscale version of the visual image. It intentionally discards the colour content in order to avoid colour/temperature ambiguities and better retain the visual textural details.

The first four values in each row of Table 6.2 are used to determine to the extent to which each pixel is deemed hot or cold. This is a fuzzy membership based on two limits. Assuming normalised inputs with colour intensities in the range of zero to one, all pixels whose LWIR colour component is less than $\alpha_{C1}$ are deemed completely cold. Pixels with LWIR intensities between $\alpha_{C1}$ and $\alpha_{C2}$ are partially cold, linearly decreasing from cold at $\alpha_{C1}$ to not cold at $\alpha_{C2}$. All pixels with LWIR intensities above $\alpha_{C2}$ are not cold. The hot fuzzy membership function works similarly but swapped with pixels above $\alpha_{H2}$ being deemed hot and below $\alpha_{H1}$ not being considered hot. Equations 6.2 and 6.3 express this mathematically:

$$f_H = \begin{cases} 0 & \text{if } \bar{I}^s_{(r,g,b,i)}.i \in [0, \alpha_{H1}] \\ \frac{\bar{I}^s_{(r,g,b,i)}.i - \alpha_{H1}}{\alpha_{H2} - \alpha_{H1}} & \text{if } \bar{I}^s_{(r,g,b,i)}.i \in (\alpha_{H1}, \alpha_{H2}] \\ 1 & \text{if } \bar{I}^s_{(r,g,b,i)}.i \in (\alpha_{H2}, 1] \end{cases} \quad \text{and} \tag{6.2}$$

$$f_C = \begin{cases} 1 & \text{if } \bar{I}^s_{(r,g,b,i)}.i \in [0, \alpha_{C1}] \\ \frac{\alpha_{C2} - \bar{I}^s_{(r,g,b,i)}.i}{\alpha_{C2} - \alpha_{C1}} & \text{if } \bar{I}^s_{(r,g,b,i)}.i \in (\alpha_{C1}, \alpha_{C2}] \\ 0 & \text{if } \bar{I}^s_{(r,g,b,i)}.i \in (\alpha_{C2}, 1] \end{cases} \tag{6.3}$$

where

$f_H$ = the membership function for hot elements,

$f_C$ = the membership function for cold elements,

$\alpha_{H1}$ = the lower threshold for the hot element,

$\alpha_{H2}$ = the upper threshold for the hot element,

$\alpha_{C1}$ = the lower threshold for the cold element,

$\alpha_{C2}$ = the upper threshold for the cold element,

$\bar{I}^s_{(r,g,b,i)}.i$ = the normalised LWIR intensity of the pixel.

TTGS uses the fuzzy hot and cold memberships to create false colour images. The visual colour triplet of the input pixels are converted to their greyscale equivalents. The red component of the output pixel is the greyscale value amplified by the hot fuzzy membership and the blue component is the greyscale value amplified by the cold fuzzy membership. The green channel is set such that the geometric mean of the three channels is the same as the original greyscale value. Equation 6.5 illustrates:

$$I^{s'}_{rgb} = \begin{bmatrix} \mathfrak{G}\,(1 + f_H) \\ \frac{\mathfrak{G}}{(1+f_H)(1+f_C)} \\ \mathfrak{G}\,(1 + f_C) \end{bmatrix} \tag{6.4}$$

where

$I^{s'}_{rgb}$ = the output RGB pixel,

$f_H$ = as per Equation 6.2,

$f_C$ = as per Equation 6.3,

$\mathfrak{G}$ = greyscale scale conversion of input RGB values,

$$= \begin{bmatrix} 0.2126 \\ 0.7152 \\ 0.0722 \end{bmatrix}^T \cdot \begin{bmatrix} \bar{I}^s_{(r,g,b,i)}.r \\ \bar{I}^s_{(r,g,b,i)}.g \\ \bar{I}^s_{(r,g,b,i)}.b \end{bmatrix}, \text{ and}$$

$$\bar{I}^s_{(r,g,b,i)} = \text{the input colour quadruplet pixel from Equation 6.1.}$$

#### 6.1.2.2 Thermal based hue offset

The TBHO algorithm attempts to retain colour information while simultaneously presenting thermal information. It first determines if the pixel is hot or cold by comparing the normalised LWIR intensity to ranges defined by the $\alpha$ values provided in Table 6.2. If the pixel is neither hot nor cold then the input RGB intensities are used as the output values. Should the pixel be deemed either hot or cold then each of the RGB values is first interpolated with the LWIR value using the $\kappa$ values of Table 6.2. Thereafter the pixel is converted from the RGB colour space to the Hue Saturation Value (HSV) colour space. This conversion is a standard procedure and can be found in many handbooks on image processing, such as that of Gonzalez and Woods [73]. Once in the HSV space, hot pixels are shifted towards 'warmer' colours and cold pixels shifted to the 'cooler' part of the colour spectrum. Finally the resultant HSV pixel is converted back into RGB format and output. This process in summarised in Equation 6.5:

$$I^{s'}_{rgb} = \begin{cases} f_{HtoR}\left({}^T\bar{P}_{HSV_C}\right) & \text{if } \bar{I}^s_{(r,g,b,i)}.i \in [\alpha_{C1}, \alpha_{C2}) \\ f_{HtoR}\left({}^T\bar{P}_{HSV_H}\right) & \text{if } \bar{I}^s_{(r,g,b,i)}.i \in (\alpha_{H1}, \alpha_{H2}] \\ \bar{I}^s_{(r,g,b,i)}.rgb & \text{otherwise} \end{cases} \tag{6.5}$$

where

$$I^{s'}_{rgb} = \text{the output RGB pixel,}$$

$$f_{HtoR} = \text{the function to transform HSV to RGB,}$$

$${}^T\bar{P}_{HSV_H} = f_{RtoH}\left({}^T\bar{I}_{RGB}\right) - \begin{bmatrix} 10^\circ \\ 0 \\ 0 \end{bmatrix},$$

$$\begin{aligned}
{}^{T}\bar{P}_{HSV_C} &= f_{RtoH}\left({}^{T}\bar{I}_{RGB}\right) + \begin{bmatrix} 8^\circ \\ 0 \\ 0 \end{bmatrix}, \\
f_{RtoH} &= \text{the function to transform RGB to HSV}, \\
{}^{T}\bar{I}_{RGB} &= \text{an intermediate intensity in RGB format}, \\
&= \begin{bmatrix} \kappa_1 \bar{I}^s_{(r,g,b,i)}.r + (1-\kappa_1)\,\bar{I}^s_{(r,g,b,i)}.i \\ \kappa_2 \bar{I}^s_{(r,g,b,i)}.g + (1-\kappa_2)\,\bar{I}^s_{(r,g,b,i)}.i \\ \kappa_3 \bar{I}^s_{(r,g,b,i)}.b + (1-\kappa_3)\,\bar{I}^s_{(r,g,b,i)}.i \end{bmatrix}, \text{ and} \\
\kappa_n &= \text{the n}^{\text{th}} \text{ interpolation factor.}
\end{aligned}$$

### 6.1.3 Quantitative measure of stitch accuracy

This section describes the quantitative measure of how accurately the stitching algorithm performs. The measure uses image content in the form of image features (Section 6.1.1). The standard implementations of SIFT [81] and SURF [82] as implemented by OpenCV [4] are used for this evaluation. All matched features that fell within the overlapped portions of the image and had matches of sufficient strength as per Tables 6.4 and 6.5 were used.

The stitching accuracy measure (Equation 6.6) is the RMS angular error in the stitch CF (over all pairs of matched image features) between the stitch positions corresponding to each camera's observation of the matched features. This angular error is calculated from the points on the multi-geometry stitch surface that correspond to camera image coordinates of the feature in the cameras. This can be expressed mathematically as:

$$\mathrm{E}^{\text{stitch}} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left[\cos^{-1}\left(\frac{\bar{T}^a_{rf_ir} \bullet \bar{T}^b_{rf_ir}}{\|\bar{T}^a_{rf_ir}\|\|\bar{T}^b_{rf_ir}\|}\right)\right]^2} \tag{6.6}$$

where

$$\begin{aligned}
\mathrm{E}^{\text{stitch}} &= \text{the RMS angular error between stitched points}, \\
N &= \text{the number of matched feature points, and}
\end{aligned}$$

$\bar{T}^j_{rf_ir}$ = stitch point for feature point $i$ in camera $j$ as per Equation 6.7.

Equation 6.6 requires the 3D spatial coordinate that corresponds to a camera's pixel coordinate for a given image point. It is not normally possible to determine the range to a single point given a single camera's observation of it. We can however determine which point on the multi-geometry stitch surface (Section 6.1.1) corresponds to that camera's pixel coordinate. Conceptually it is easier to think of the process in reverse: the pixel position is corrected for lens distortion and then converted into a UV using the camera's intrinsic parameters. Since the camera's pose is known relative to the stitch geometry (i.e the camera's extrinsic parameters), the UV can be extended until it intersects either the stitch plane or the stitch sphere. This point is the 3D position required. In practice, such an iterative procedure is not required as the distance to the plane and sphere can be calculated analytically and the closer point selected. Equation 6.7 shows how this is done:

$$\bar{T}^j_{rf_ir} = \bar{T}_{rc_jr} + D_{c_jf_ir}\bar{U}_{c_jf_ir} \tag{6.7}$$

where

$\bar{T}^j_{rf_ir}$ = projection of camera $j$'s feature $i$ onto the stitch surface,

$D_{c_jf_ir}$ = distance from camera $j$ to feature $i$'s stitch surface projection,

$= \min(D^S_{c_jf_ir}, D^P_{c_jf_ir})$,

$D^S_{c_jf_ir}$ = distance from camera $j$ to the sphere for feature $i$,

$$= \max\left[\left(-\left(\bar{T}_{rc_jr} \bullet \bar{U}_{c_jf_ir}\right) \pm \sqrt{\left(\bar{T}_{rc_jr} \bullet \bar{U}_{c_jf_ir}\right)^2 - (\|\bar{T}_{rc_jr}\|^2 - D^2_{rsr})}\right)\right],$$

$D^P_{c_jf_ir}$ = distance from camera $j$ to plane feature $i$,

$$= \begin{cases} \frac{D_{pr}+\bar{T}_{rc_jr}\bullet\bar{U}_{rpr}}{\bar{U}_{c_jf_ir}\bullet\bar{U}_{rpr}} & \text{if } \left(\bar{U}_{c_jf_ir} \bullet \bar{U}_{rpr}\right) > 0 \\ 2D_{rsr} & \text{if } \left(\bar{U}_{c_jf_ir} \bullet \bar{U}_{rpr}\right) \leqslant 0 \end{cases},$$

$D_{pr}$ = distance to the plane from the reference origin,

$D_{rsr}$ = distance from the reference to the sphere, i.e the stitch radius,

$\bar{U}_{rpr}$ = outward facing normal UV of the plane,

$$
\begin{aligned}
\bar{U}_{c_j f_i r} &= \text{feature } i\text{'s direction vector in } j\text{'s CF in reference CF,}\\
&= \mathbf{R}_{c_j r}\mathfrak{f}^{\text{img}\to\text{vec}}(\mathfrak{f}^{\text{Brown}}(\bar{P}^d_{i,j}, \bar{DU}_{\text{params},j}), \bar{I}_{params,j}),\\
\mathbf{R}_{c_j r} &= \text{rotation matrix to convert CFs from } j\text{'s to the reference CF,}\\
\bar{P}^d_{i,j} &= \text{(distorted) pixel coordinate of feature } i \text{ in camera } j\text{'s image,}\\
\bar{DU}_{\text{params},j} &= \text{camera } j\text{'s distortion correction parameters,}\\
\bar{I}_{params,j} &= \text{camera } j\text{'s intrinsic parameters,}\\
\mathfrak{f}^{\text{img}\to\text{vec}} &= \text{as per Equation 3.17 (Section 3.4.2 for parameter description),}\\
\mathfrak{f}^{\text{Brown}} &= \text{as per Equation 3.25 (Section 5.5.1 for parameter details), and}\\
\bar{T}_{rc_j r} &= \text{position of camera } j \text{ in the reference CF.}
\end{aligned}
$$

The calculation of the point on the stitch sphere corresponding to the (non-centrally located) camera's UV associated with its observation of that point, is merely another application of the cosine rule. The three sides of the triangle are the stitch radius, the camera translation from the stitch origin, and the unknown distance from the camera to the stitch surface point. It is then possible to calculate the angle opposite the radius from the camera's UV to the point and the translation vector of the camera. Equation 6.8 shows, with reference to Figure 3.9, how this is done taking into account the necessary CFs:

$$a^2 = b^2 + c^2 - 2bc\cos(\hat{A}) \tag{6.8}$$

where

$$
\begin{aligned}
a &= \text{the known stitch radius i.e } D^s,\\
b &= \text{the magnitude camera displacement w.r.t. the stitch CF,}\\
&= \|\bar{T}_{c_j r c_j}\|,\\
c &= \text{the desired distance of the stitch point from the camera,}\\
&= D^S_{c_j f_i r},\\
\cos(\hat{A}) &= \text{cosine of the angle opposite side } a,\\
&= \text{cosine of the angle between camera's translation and feature vectors,}\\
&= \frac{1}{\|\bar{T}_{c_j r c_j}\|}\bar{T}_{c_j r c_j} \bullet \bar{U}_{c f_i c},
\end{aligned}
$$

$$\begin{aligned}\bar{T}_{c_j r c_j} &= \text{the displacement of reference origin from camera } j,\\ \bar{U}_{c_j f_i c_j} &= \text{the vector from camera } j \text{ to feature } i,\\ i &= \text{the feature number, and}\\ j &= \text{the camera number.}\end{aligned}$$

Equation 6.8 can be rearranged to solve for $c$. The angle between $\bar{T}_{rc_jc_j}$ and $\bar{U}_{c_jf_ic_j}$ is equal to $\pi - \hat{A}$. However, since $\cos(\pi - \theta) = -\cos(\theta)$ and $\bar{T}_{c_jrc_j} = -\bar{T}_{rc_jc_j}$, the sign effects cancel. The resulting quadratic equation has real roots if the camera is inside the stitch sphere. The positive root is where the camera's ray intersects the sphere in front of the camera. The negative root is where the ray intersects behind the camera and can be ignored. Ergo the maximum is taken in Equation 6.7. Equation 6.7 also expresses $\bar{T}_{c_jrc_j}$ and $\bar{U}_{c_jf_ic_j}$ in the reference CF.

Equation 6.7 uses the DU parameters to calculate the UV associated with a pixel position, and Equation 6.1 uses the UD parameters to convert a vector to a pixel position. Neither of the two mappings between the distorted and undistorted domains in either direction is perfect. Thus converting from the distorted domain to the undistorted domain and then back to the distorted domain will not perfectly result in the original point, i.e.

$$\bar{P}^d \neq \mathfrak{f}^{\text{Brown}}\left(\mathfrak{f}^{\text{Brown}}(\bar{P}^d, \bar{DU}_{\text{params}}), \bar{UD}_{\text{params}}\right). \tag{6.9}$$

The same is true of converting from the undistorted domain to the distorted domain and back. This means that the point on the stitch surface calculated by Equation 6.7 will not result in the same pixel position for the feature when fed through Equation 6.1. This may be countered by numerically refining the undistorted pixel position such that the resultant undistorted position does indeed produce the original distorted position when corrected with the DU parameters, thus providing a measurement of the stitch accuracy that better reflects how the stitching is implemented in practice. This removal of the DU error does not desensitise the stitch to the DU calibration, since the DU calibration was used to determine the camera w.r.t. the mount and mount w.r.t. the reference poses as well as the focal lengths. Equation 6.10 shows how the refined distorted position can be calculated to provide a better direction vector for each feature (i.e $\bar{U}_{c_j f_i r}$) in Equation 6.7:

$$\bar{P}^{u*}_{i,j} = \underset{\bar{P}^{u}_{i,j}}{\operatorname{argmin}} \left[ \|\bar{P}^{d}_{i,j} - \mathfrak{f}^{\text{Brown}}(\bar{P}^{u}_{i,j}, \bar{UD}_{\text{params,j}})\|^2 \right] \tag{6.10}$$

where

$\bar{P}^{u*}_{i,j}$ = the refined undistorted pixel position corresponding to $\bar{P}^{d}_{i,j}$,

$\bar{P}^{u}_{i,j}$ = estimate of undistorted pixel position corresponding to $\bar{P}^{d}_{i,j}$,

$\bar{P}^{d}_{i,j}$ = the input distorted pixel position,

$\mathfrak{f}^{\text{Brown}}$ = as per Equation 3.25 (Section 3.4.6 for parameter details),

$\bar{UD}_{\text{params,j}}$ = camera $j$'s DU parameters,

$i$ = the feature number, and

$j$ = the camera number.

### 6.1.4 Stitching and fusion results

This section provides the pictorial and quantified accuracy (Section 6.1.3) results of the stitching (Section 6.1.1) and fusion (Section 6.1.2) algorithms described earlier.

Results are provided for two systems, details of which are given in Table 6.3. The systems had different numbers of cameras with different resolutions, pixel formats, and imager sizes. The FOVs differed between the systems both in terms of the overall system FOV and the individual FOVs. System 1, shown in Figure 6.2, had two visual cameras vertically displaced above two LWIR cameras. System 2 was designed with berths for 5 camera pairs, each consisting of one visual and one LWIR camera. However, for cost reasons only 3 LWIR cameras were used (in the three central berths) resulting in only 8 cameras in the system. The camera array of System is shown in Figure 6.3. The LWIR HFOV of System 2 is thus less than its visual HFOV, whereas the LWIR Vertical Field of View (VFOV) is larger than the visual VFOV

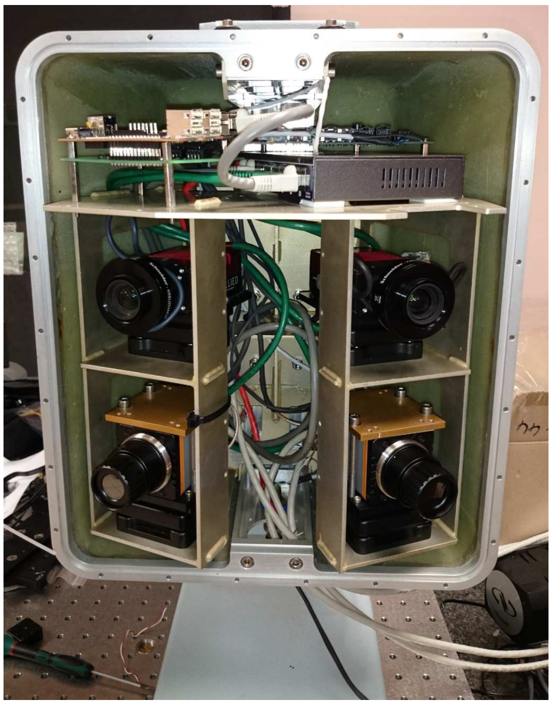


Figure 6.2: Inner camera array assembly of System 1.

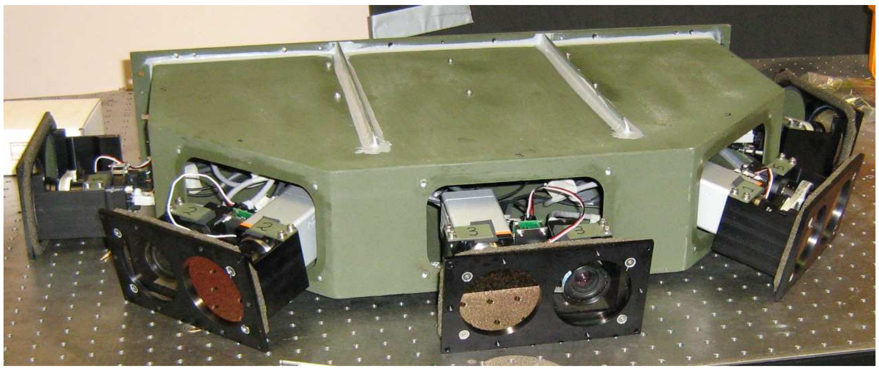
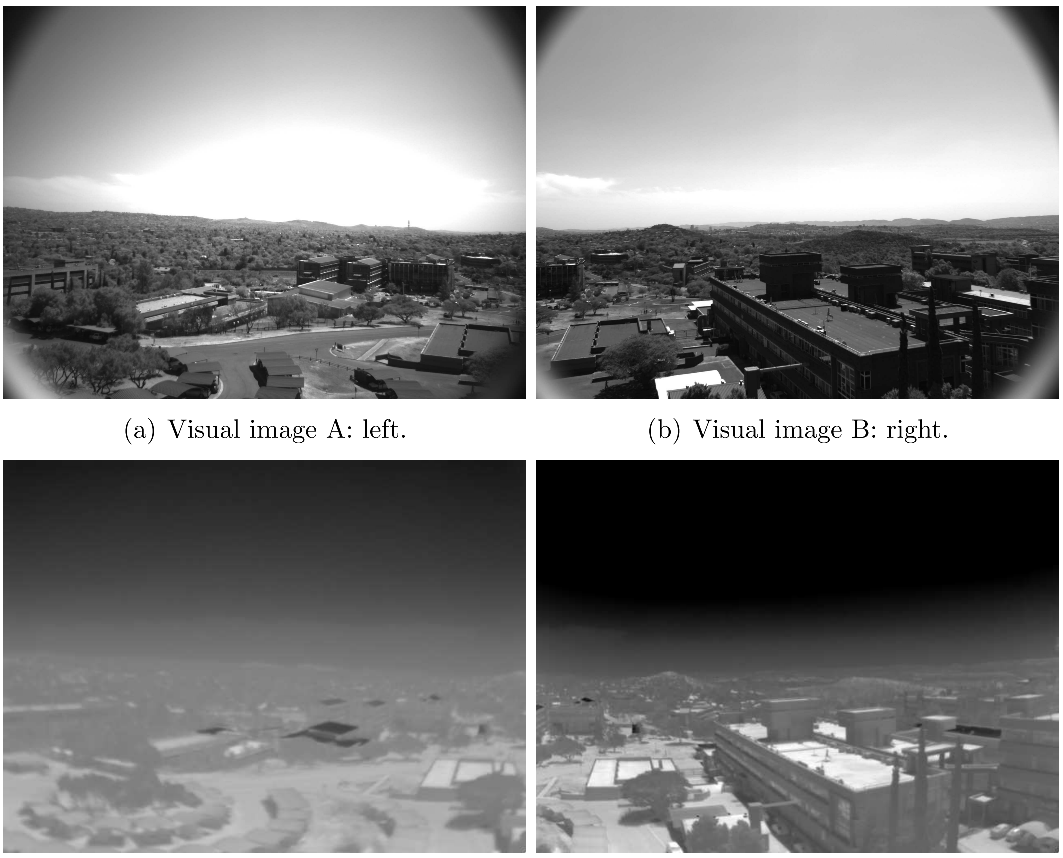


Figure 6.3: Partially disassembled structure of System 2 showing the cameras.

Table 6.3: Stitch system physical specifications.

| | | **System 1** | **System 2** |
|---|---|---|---|
| Visual cameras | Number | 2 | 5 |
| | Format | Greyscale | Colour, Bayer |
| | Resolution | 1936 × 1456 | 1280 × 960 |
| | Focal length | 8.0 mm | 5.0 mm |
| | Imager size | 2/3” | 1/3” |
| LWIR cameras | Number | 2 | 3 |
| | Format | Greyscale | |
| | Resolution | 640 × 480 | |
| | Focal length | 10 mm | |
| | Pixel size | 17 $\mu$m×17 $\mu$m | |
| System | Total camera tally | 4 | 8 |
| | Array format | 2 × 2 | 8 × 1 |
| | Visual HFOV | 100° | 200° |
| | Visual VFOV | 38° | 30° |
| | LWIR HFOV | 100° | 120° |
| | LWIR VFOV | 38° | 40° |
| | Stitch FOV | 100° × 47.5° | 220° × 49° |
| | Stitch radius | 500 m | 150 m |
| | Stitch plane distance | 20 | 2.5 m |
| | Stitch plane normal | (0, 0, -1) | (0, 0, -1) |
| | Stitch resolution | 1600 × 760 | 1800 × 400 |

The pictorial results include the original distorted images (Figures 6.4, 6.11 and 6.12), visual stitches (Figures 6.5 and 6.13), LWIR stitches (Figure 6.6 and 6.14) and fused stitches (Figures 6.7, 6.15 and 6.16) for the two systems. The quantified accuracy results are the stitch accuracy for each overlapping adjacent camera pair of the same spectral frequency. Inter-band statistics are not provided as image feature matching across spectral bands were shown by Cronje and de Villiers [13] to perform poorly. This is because the image content differs (thermal signature versus visual texture). Even if the same object is identified as a feature in the both the visual and LWIR images, its description is sufficiently different that matching is impossible [13]. Statistics for the raw feature matching results as (Equation 6.6) and the refined results (Equation 6.10) are presented for both SIFT [81] and SURF [82] features. These statistical results are presented in both



(a) Visual image A: left.
(b) Visual image B: right.
(c) LWIR image A: left.
(d) LWIR image B: right.

Figure 6.4: System 1 input images.

tabular form (Tables 6.4 and 6.5) and as box plots (Figures 6.9 and 6.19).

Figure 6.4 provides examples of raw images captured by System 1. These images were manually captured synchronously. Severe vignetting induced by the lenses is visible in Figures 6.4(a) and 6.4(b). Compared to Figure 6.4(d), the focus of Figure 6.4(c) is noticeably softer. It was not possible to focus the LWIR camera to infinity. The LWIR images had both a lower resolution (Table 6.3) and wider FOV as indicated by additional carports being evident on the bottom left of Figure 6.4(c) compared to 6.4(a) and more of the second building being in visible in the right of Figure 6.4(d) compared to 6.4(b).

Figures 6.5, 6.6 and 6.7 provide visual-only, IR-only and fused stitches respectively. There is slight blurring evident in the seam of the visual stitch (Figure

6.5), as expected. Both the far and near sections of scene are stitched well. This is due to the stitch plane coinciding well with the ground in the scene. The visual-only and IR-only images show that cameras of the same spectrum can successfully be stitched. Figure 6.7 shows that these stitches are of the same scale, orientation, and displacement and that the corresponding points between the visual and LWIR spectra are correctly registered. As the input images were already greyscale, only the TTGS fused stitch is presented.

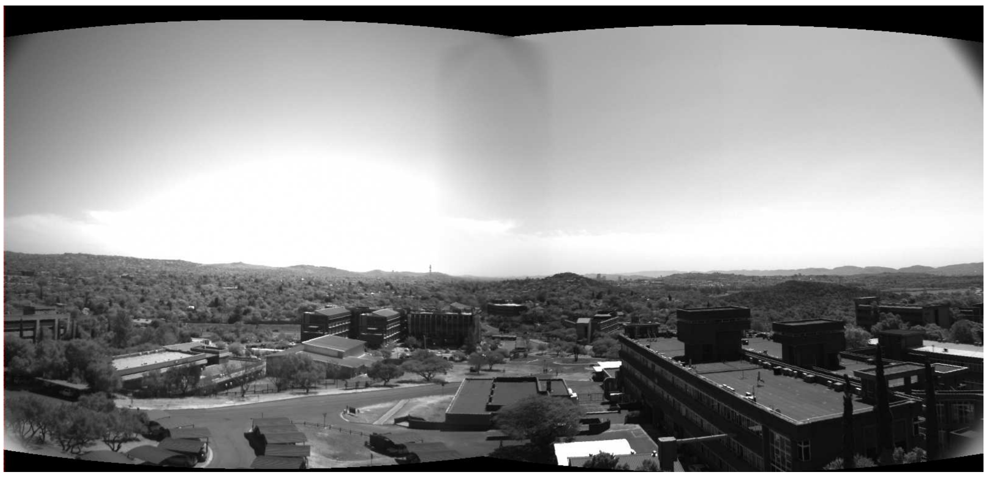


Figure 6.5: Visual stitch from System 1.

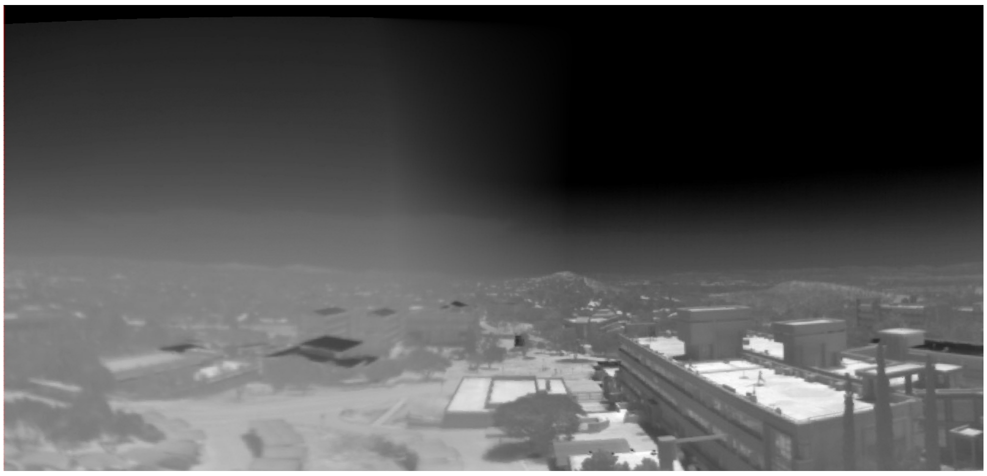


Figure 6.6: IR stitch from System 1.

In order to quantify the stitching accuracy, SIFT [81] and SURF [82] features were found in the overlapping regions of the visual pair and the LWIR pair. Figure

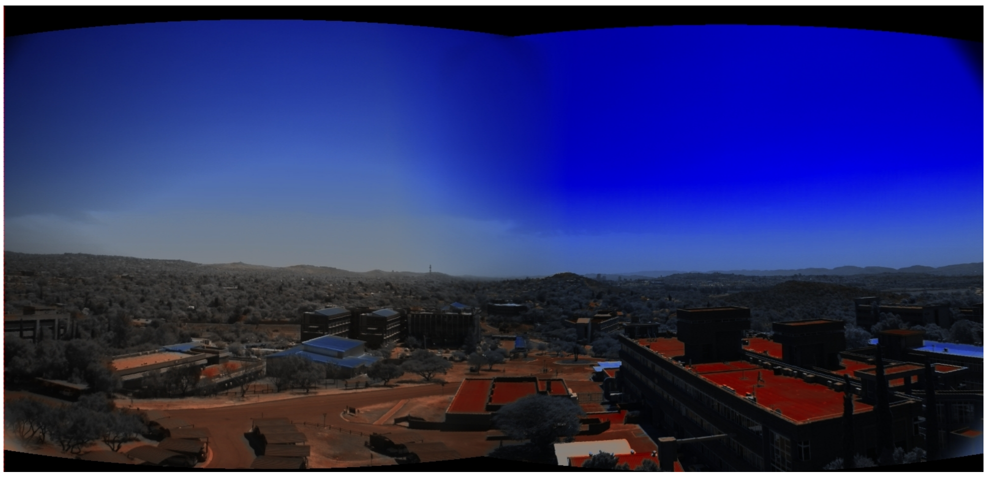
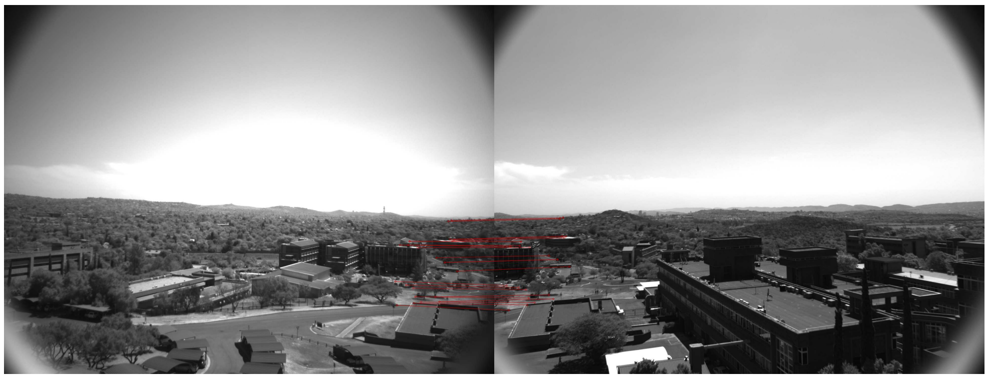


Figure 6.7: TTGS fused visual and LWIR stitch from System 1.

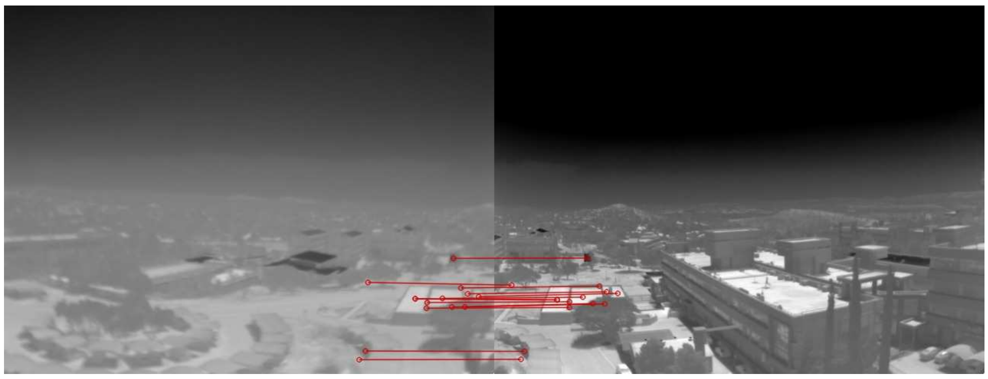
6.8 shows the SIFT features that were found and successfully matched. Only 'strong' features meeting the minimum Hessian for detection, and description distances as defined by Table 6.4 are plotted. For the sake of brevity, the SURF feature matches are not provided.

Each of the features detected by SIFT/SURF were projected onto the stitch surface using the techniques described in Section 6.1.3. The error (in degrees) in the stitched image between these points was then calculated. Table 6.4 provides the results of the statistical analysis of these stitch errors for both the raw and refined projections (see Section 6.1.3). Outliers were defined as points that lie more than 2.5 standard deviations away from the mean, as calculated from the inlier population. The values are very acceptable, showing both mean and median values in he range of $0.085°$ to $0.1°$ for both the raw and refined results for both sets of features.

Figure 6.9 visually depicts the statistical data given in Table 6.4. The box plots extend from the 25th to the 75th percentiles with the median marked in between. The whiskers are symmetrical about the mean and extend one standard deviation from it. The points beyond the whiskers are the outliers. The outliers are excluded from the statistical analysis. The box plots have a logarithmic horizontal axis to facilitate the simultaneous visualisation of the tightly grouped



(a) Visual matches between Figures 6.4(a) and 6.4(b).



(b) LWIR matches between Figures 6.4(c) and 6.4(d).

Figure 6.8: System 1 SIFT matches.

Table 6.4: Feature based stitch errors for System 1.

| | **SURF** | | **SIFT** | |
|---|---|---|---|---|
| | **Raw** | **Refined** | **Raw** | **Refined** |
| Minimum Hessian | 300 | | 2000 | |
| Maximum descriptor distance | 0.1 | | 200 | |
| Valid features | 122 | | 37 | |
| Mean (degrees) | 0.086 | 0.085 | 0.093 | 0.1002 |
| Standard deviation (degrees) | 0.028 | 0.031 | 0.048 | 0.060 |
| 25th percentile (degrees) | 0.068 | 0.066 | 0.059 | 0.059 |
| Median (degrees) | 0.084 | 0.082 | 0.085 | 0.082 |
| 75th percentile (degrees) | 0.103 | 0.105 | 0.110 | 0.127 |
| Outliers | 14 | 11 | 5 | 4 |

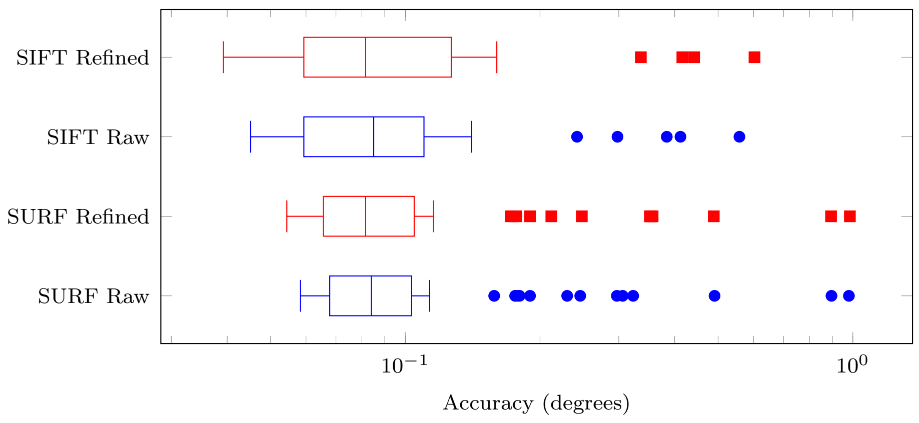


Figure 6.9: Stitching accuracy box plots for System 1.

inlier population and the distant outliers, which range in value from 0.15° to 0.35°.

The stitch coordinates of the outlier pairs were plotted on the stitched image in order to determine if the outliers were truly anomalous or indicative of an underlying error. This is shown in Figure 6.10 where the combined refined outliers from both SIFT and SURF and both both the visual and LWIR spectrums. There is no systemic nature noticeable in the outliers.

The outliers in the sky and the uniform grass areas in the middle of the image are likely to just be erroneous matches. The same is true of the ones in the tree, which has a repetitive texture. Finally, the two outliers on the left have significant vertical components that are unlikely to be due to errors produced by horizontally splayed cameras.

System 2 consists of two interlaced linear arrays, one array of 5 visual cameras and another of 3 LWIR cameras. The LWIR cameras cover approximately the same FOV as the three central visual cameras. Like the images of System 1, System 2's images were recorded simultaneously and so there is no time discrepancy between the images. Figure 6.11 shows example visual images from left to right (w.r.t. the cameras' point of view). Figure 6.12 shows corresponding LWIR images also ordered from left to right.

Of interest is the lens flares in Figures 6.11(d) and 6.11(e). Note how the contrast of the building in their overlap is different in the two images, and how the sky appears different in these images compared to Figure 6.11(c). Of further interest is the soft focus of the central LWIR camera shown in Figure 6.12(b). Note how different the overlapped region of Figures 6.12(a) and 6.12(b) appears.

Table 6.1 provides the stitching parameters for System 2. Figure 6.13 provides the visual stitch for System 2. Figure 6.14 provides the LWIR stitch. The fused stitch images are provided by Figures 6.15 and 6.16 which show the colour preserving TBHO algorithm and the greyscale TTGS algorithm fusion results respectively.

Note how the extremely close container in the overlap between 6.11(d) and 6.11(e) is stitched fairly well. Only a minor discontinuity is evident on the closer edge of the roof (Figures 6.13 through 6.16) while the more distant building in the same overlap is stitched with no apparent errors. This is due to the container being significantly closer than the stitch distance. The shadows beneath the container in the overlap of Figures 6.11(c) and 6.11(d) in the visual spectrum and Figures 6.12(b) and 6.12(c) in the LWIR spectrum are stitched fairly well, despite being much closer than the stitch sphere radius. This is because they lie near the stitch plane which was defined to closely coincide with the ground in the scene (Table 6.1). In contrast, the window of the container is distinctly blurred. This

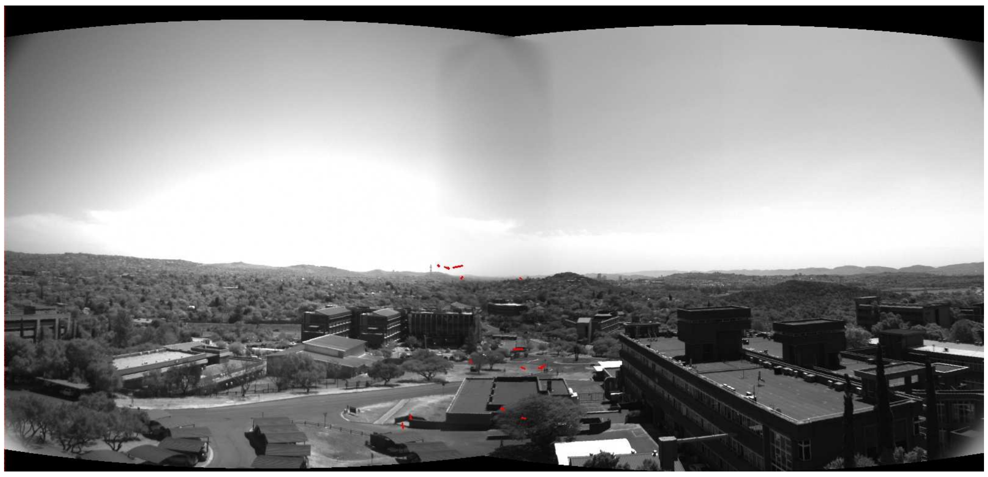
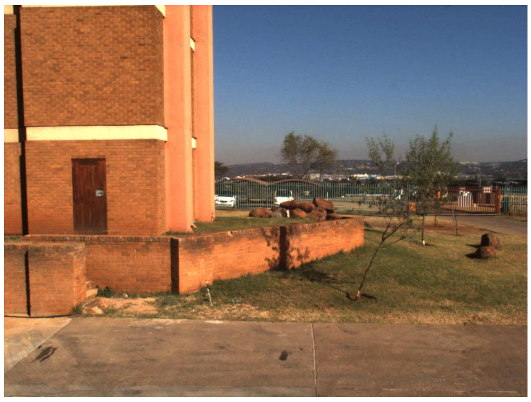
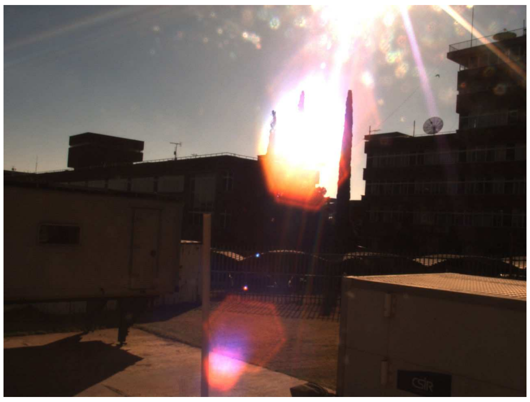


Figure 6.10: SIFT and SURF outlier feature pairs for System 1.


(a) Visual image A: outer left.

(b) Visual image B: inner left.

(c) Visual image C: centre.


(d) Visual image D: inner right.

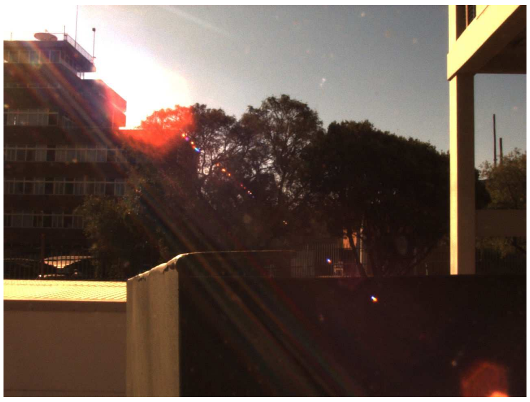
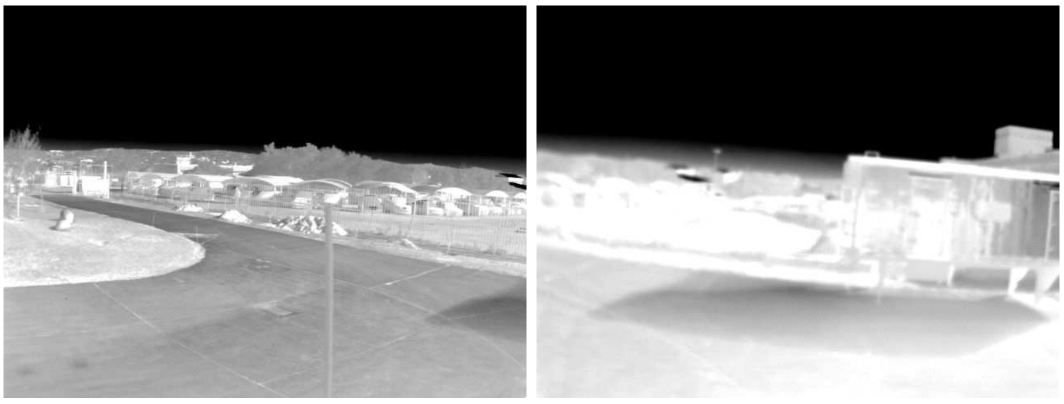

(e) Visual image E: outer right.

Figure 6.11: System 2 visual input images.


(a) LWIR image A: left. (b) LWIR image B: centre.

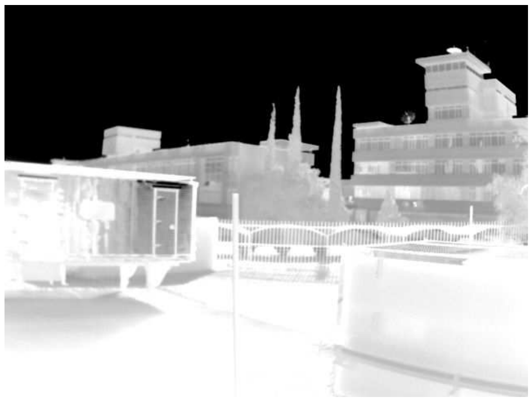

(c) LWIR image C: right.

Figure 6.12: System 2 LWIR input images.

is because the window is both off the ground plane and much closer than the stitch radius. The grass verge and rock in the overlap Figures 6.11(c) and 6.11(d) are also stitched well despite their proximity to the cameras, because they too lie near the predefined stitch plane.

It is also worth noting that despite the soft focus of the central LWIR camera, lens flares and poor colour balancing of the two rightmost visual cameras, the stitches were aligned correctly (in the far field). Lens flares and solar effects are increasingly unavoidable as more cameras are added or as the solid angle that the stitch subtends is increased. Thus the image content independence, which is a characteristic of photogrammetric stitching, is vital.

Figure 6.17 shows the SIFT features (the SURF features are again not presented

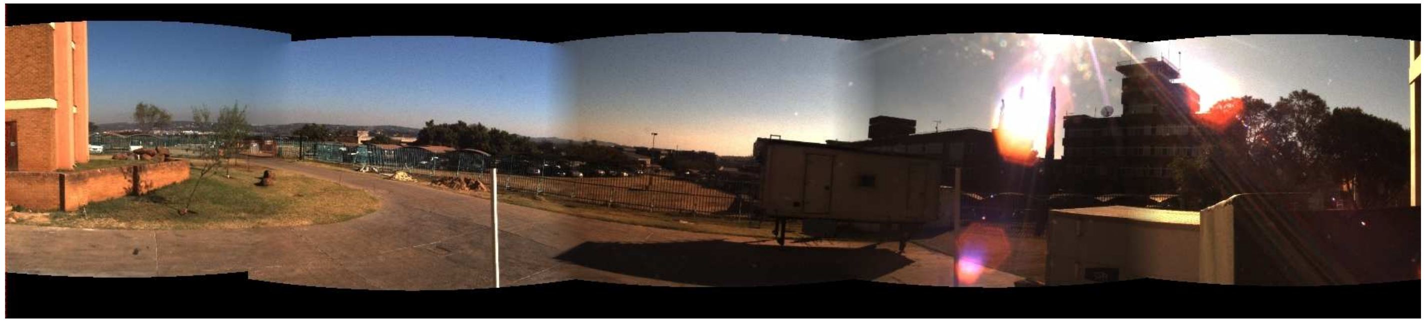


Figure 6.13: Visual stitch from System 2.

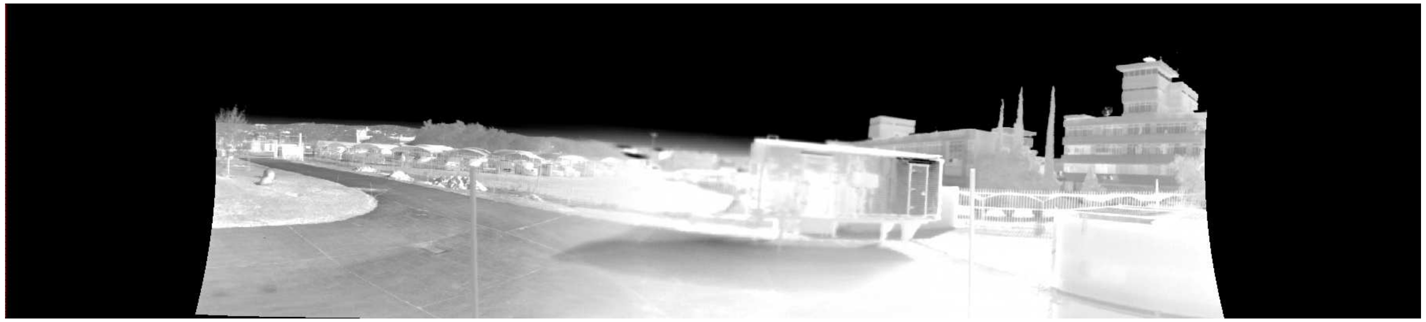


Figure 6.14: LWIR stitch from System 2.

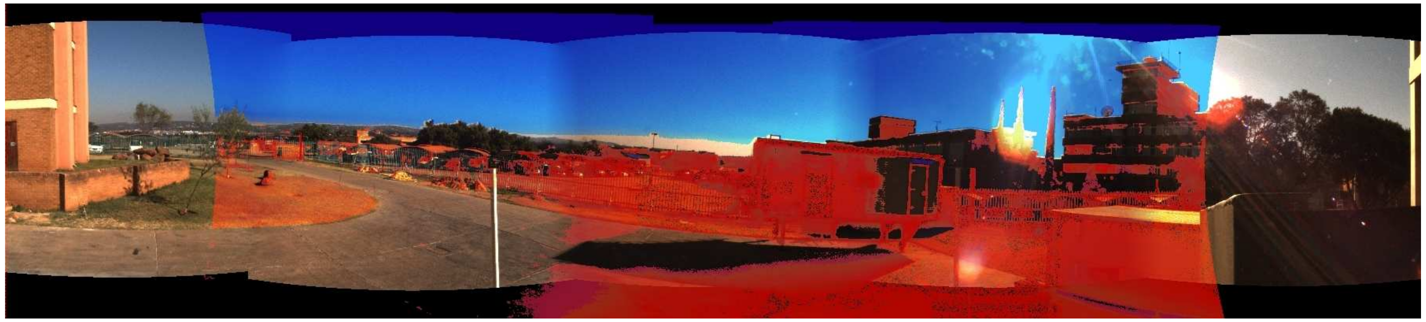


Figure 6.15: Dual-band TBHO fused stitch from System 2.

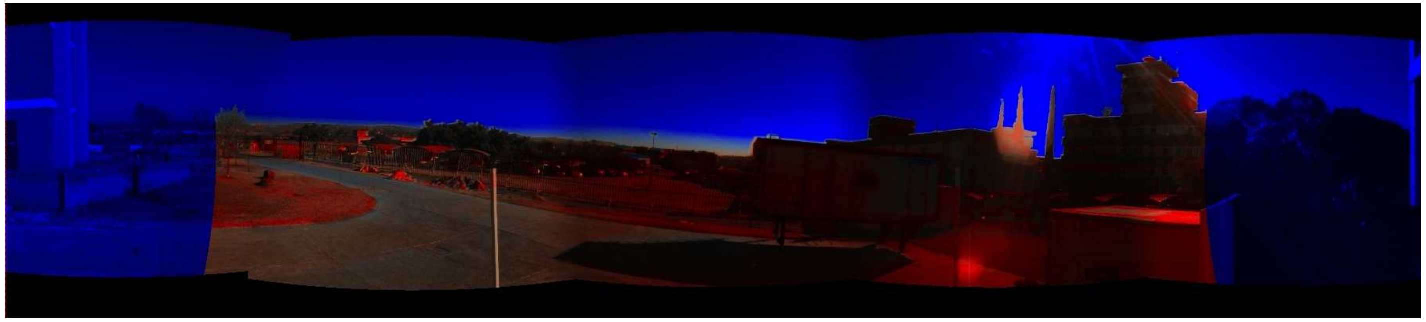


Figure 6.16: Dual-band TTGS fused stitch from System 2.

Table 6.5: Feature based stitch errors for System 2.

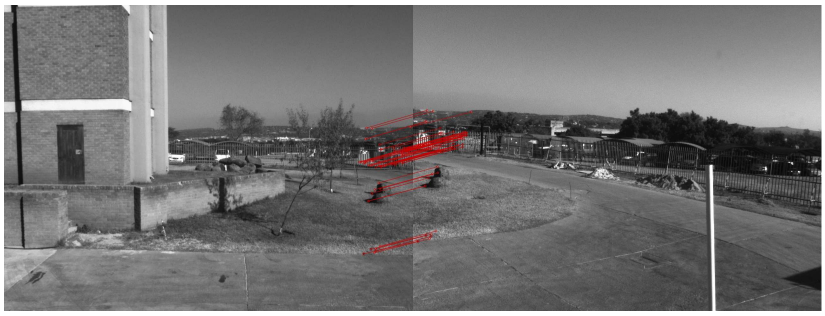
| | **SURF** | | **SIFT** | |
|---|---|---|---|---|
| | **Raw** | **Refined** | **Raw** | **Refined** |
| Minimum Hessian | 300 | | 2000 | |
| Maximum descriptor distance | 0.1 | | 150 | |
| Valid features | 85 | | 133 | |
| Mean (degrees) | 0.203 | 0.201 | 0.273 | 0.272 |
| Standard deviation (degrees) | 0.137 | 0.138 | 0.139 | 0.140 |
| 25th percentile (degrees) | 0.087 | 0.084 | 0.175 | 0.176 |
| Median (degrees) | 0.166 | 0.165 | 0.272 | 0.269 |
| 75th percentile (degrees) | 0.347 | 0.347 | 0.375 | 0.376 |
| Outliers | 9 | 9 | 9 | 9 |

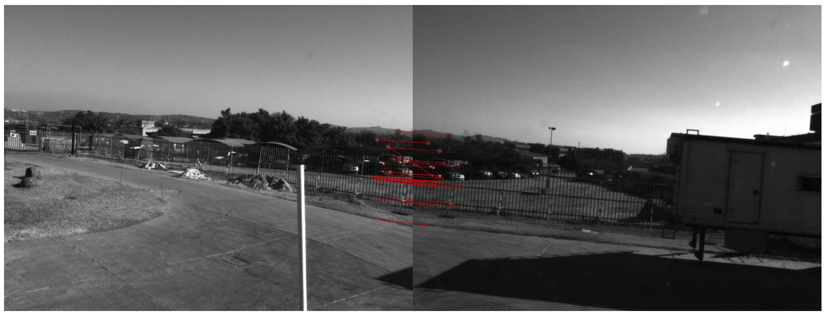
for brevity) for the four pairs of adjacent visual cameras of System 2. Figure 6.18 shows the SIFT features for System 2's LWIR camera pairs. It is unsurprising to note how the number of matched features decreases drastically when one or both images has lens flares, poor colour balance, or soft focus. For instance compare the number of SIFT features marked in Figure 6.17(a) and 6.17(b) to those marked in Figures 6.17(c) and 6.17(d). This again highlights the potentially fallibility of feature based methods in uncontrolled environments.

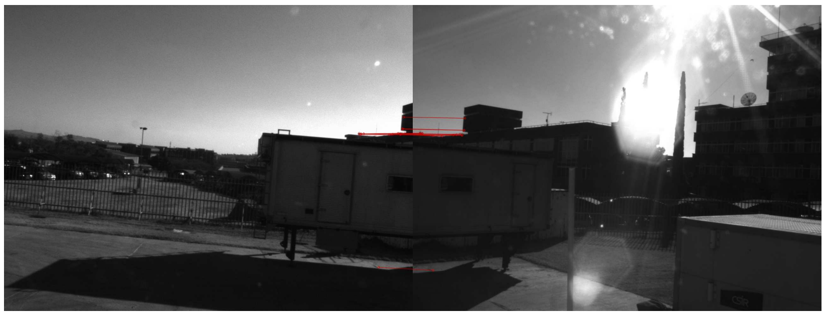
Table 6.5 presents the same statistical analysis results for feature pairing of System 2 that Table 6.4 does for System 1. The mean and median values for both the SIFT and SURF feature sets are on the order of $0.2°$ to $0.3°$. SIFT found both more features and had a higher variance of errors. SIFT also had fewer outliers despite having more points.

Figure 6.19 shows the box plots for the analysis of System 2. The interpretation of Figure 6.19 is the same as that of Figure 6.9. It is evident that both SIFT and SURF (for both raw and refined projections) had a cluster of outliers between $0.6°$ and $1.2°$. These outliers feature pairs were converted into the stitch coordinates and plotted in Figures 6.20 and 6.21 for SIFT and SURF features, respectively. Outliers drawn in blue are from visual cameras and red are from LWIR cameras.

It is apparent that the majority of the smaller outlier features (i.e shorter lines in Figures 6.20 and 6.21) are in the near field of the stitch and are thus attributable


(a) Matches between Figures 6.11(a) and 6.11(b).


(b) Matches between Figures 6.11(b) and 6.11(c).


(c) Matches between Figures 6.11(c) and 6.11(d).

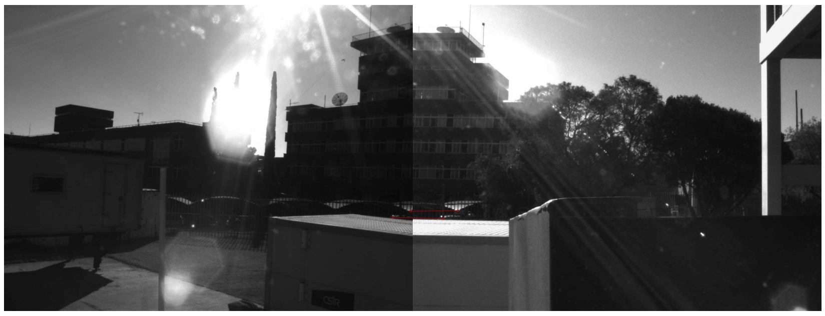
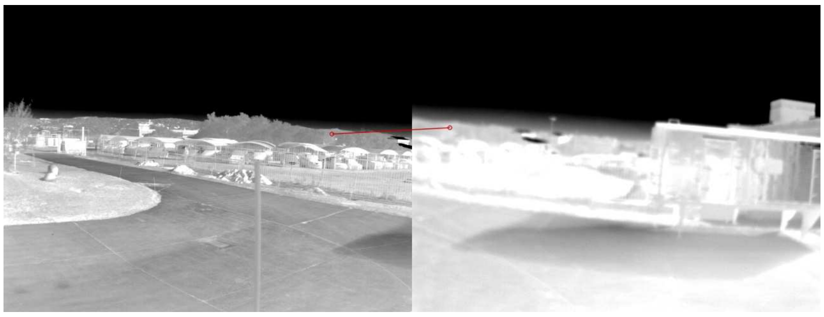
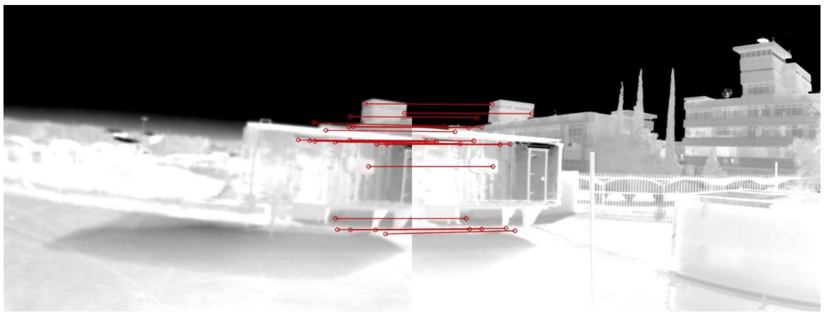

(d) Matches between Figures 6.11(d) and 6.11(e).

Figure 6.17: System 2 SIFT visual matches.



(a) Matches between Figures 6.12(a) and 6.12(b).



(b) Matches between Figures 6.12(b) and 6.12(c).

Figure 6.18: System 2 SIFT LWIR matches.

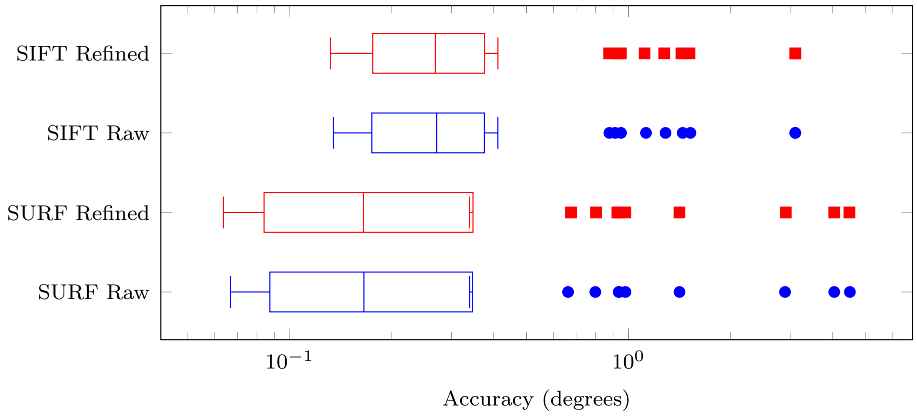


Figure 6.19: Stitching accuracy box plots for System 2.

to the known depth sensitivity. Several outliers are also clustered on the palisade fence which could be due to incorrect matching between similar repetitive features. The larger outliers may all be disregarded as they have a significant vertical component which is unlikely to be caused by a horizontal linear array of cameras and so are likely erroneous feature matches.

### 6.1.5 Stitching conclusion

Stitched panoramas were successfully created from cameras arranged in different physical configurations as both 1D and 2D arrays. The cameras had different resolutions, focal lengths, pixel sizes, CCD dimensions, FOVs, and sensitivity spectra.

Subjective evaluation of the stitched panoramas suggests the stitching is performed correctly for far field objects. This is evidence that the camera calibration routines of Chapter 5 provide results with real world applicability and usefulness. The near field objects were not stitched correctly, this is a consequence of the stitching algorithm's depth sensitivity and not of the camera calibration accuracy. Of course, if the stitch distances were decreased, the near field objects could be stitched correctly, at the expense of the far field objects. The alignment between cameras of the same spectrum and across spectrums is good.

The stitch sensitivity to depth was drastically improved by moving from a single to a multi-geometry stitch, the details of this were published in 2013 [12]. The remaining stitch sensitivity may then be exploited by performing the stitching at different distances and choosing the best distance for each pixel. This method is similar to the planes-sweep methods [87, 88] and is a planned future upgrade to the current stitch systems.

A quantitative evaluation based on image features shows that the stitching errors (Tables 6.4 and 6.5) are on the order $0.08^\circ$ to $0.3^\circ$ for the two systems. For both systems, SIFT features were more abundant and yielded slightly worse results than SURF features. At the stitch resolutions given in Table 6.3 these angles equate to mean errors in the order 3.7 to 6.1 pixels for System 1 and 1.4 to to

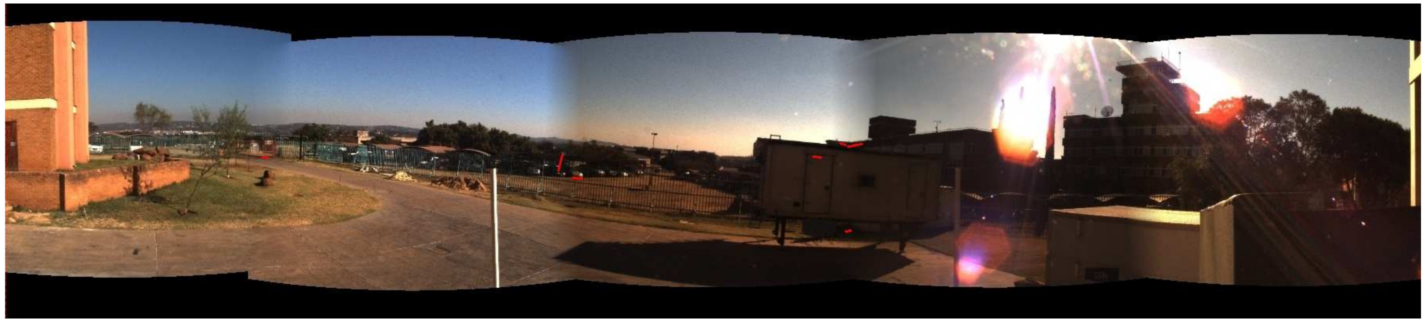


Figure 6.20: SIFT outlier feature pairs for System 2.

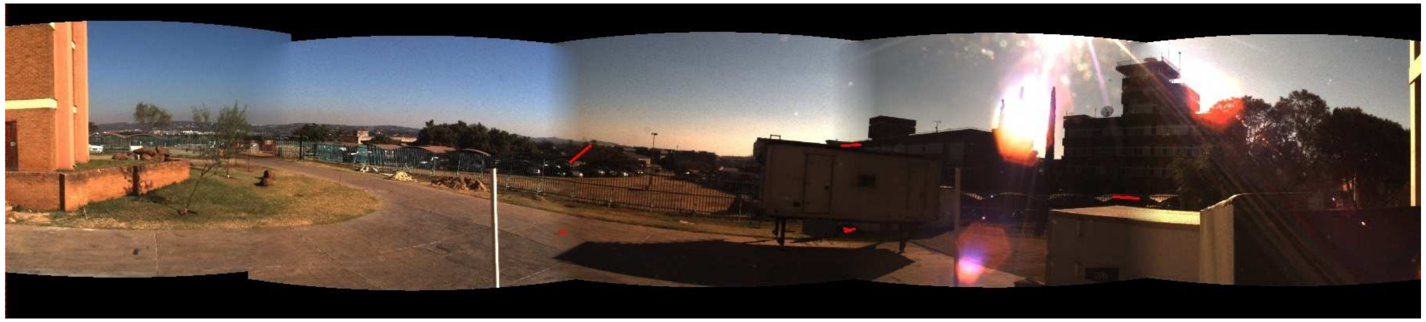


Figure 6.21: SURF outlier feature pairs for System 2.

2.2 pixels for System 2.

It is worth recalling that the SIFT and SURF features are all found in the periphery of the images (i.e. the overlapping regions) where the residual distortion error is typically the highest [15]. This means that if the cameras had a higher overlap the error would decrease. This statement is further supported by the subjectively good alignment between the 100% overlapped LWIR and visual cameras. Secondly, the blending system employed will mitigate the apparent effect of these minor discrepancies as the camera which has the point closest to its centre will have a higher weighting factor. It is thus reasonable to say that the quantified analysis also supports the conclusions that the camera calibration routines of Chapter 5 are suitable for stitching purposes.

The examples presented here were chosen to highlight the robustness of photogrammetric stitching and to allow for quantification of the stitching accuracy. The amount of overlap between adjacent images is excessive for photogrammetric stitching (indeed no overlap at all is required) and was only made so large to allow for sufficient image features to be present to quantify the stitch accuracy.

Similarly no effort was made at correcting the image exposure and focus of the systems. While this is indisputably vital in a deployed system it has no bearing (for telecentric lenses at least) on the photogrammetric aspects which are the subject of this thesis. Xu [89] provides an overview of recent techniques to perform sharpness and colour equalisation across multiple live overlapping video feeds.

## 6.2 Optical helmet tracking

A laboratory demonstration version of an optical helmet tracker was developed to further prove the applicability of the APCCS. A summary of the work is presented here, further details can be found in the paper presented in 2014 [7].

An ideal 'helmet' (from a tracking perspective) was created. The helmet was cuboid in shape made from rigid metal and had a tetrahedral cluster of LEDs on four sides. The fifth side contained a mounting interface to the robotic arm and the sixth side (the one opposite the arm mounting interface) was left unpopulated. Figure 6.22 shows the "helmet" mounted on the robot arm. The LEDs were controlled by an Arduino Uno using an "RGB LED Matrix Adapter Shield" to provide independent current sources to control the brightness of each LED. Communications were via a "Sparkfun Bluetooth Mate Silver" which is visible in Figure 6.22. The 3D spatial positions of the LEDs on the helmet were measured relative to the robot end effector using a method similar to that of the calibration LED spatial offset described in Section 5.4.2.

Two cameras with different resolutions and lenses with different focal lengths were used for the evaluation as seen in Figure 6.22. The red camera is a 3 megapixel AVT GT1920 with an 8 mm focal length Schneider Cinegon 1.4/8-

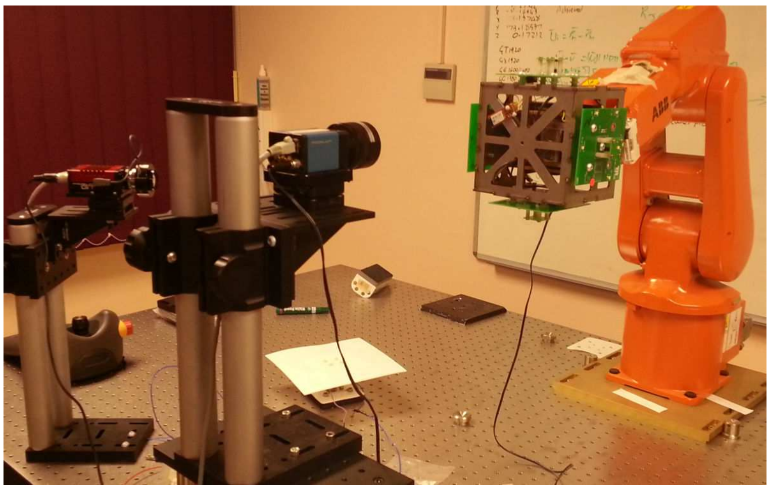


Figure 6.22: Laboratory optical helmet tracker apparatus.

Table 6.6: Helmet tracker accuracy test results.

| | **Measurement** | **Unit** | **System** | **Camera 1** | **Camera 2** |
|---|---|---|---|---|---|
| Angular | Mean | degrees | 0.137 | 0.103 | 0.188 |
| | Standard deviation | | 0.053 | 0.028 | 0.038 |
| | RMS | | 0.147 | 0.107 | 0.192 |
| | RMS | mrad | 2.566 | 1.868 | 3.351 |
| Translation | Mean | mm | 6.286 | 6.494 | 5.973 |
| | Standard deviation | | 0.446 | 0.380 | 0.673 |
| | RMS | | 6.302 | 6.506 | 6.011 |

1902 lens. The blue camera is a 2 megapixel AVT GE1600 with a 4.8 mm focal length Schneider Cinegon 1.8/4.8-0302 lens.

This simple system required 112 parameters to be determined: 33 parameters per camera (Table 6.1) and another 48 parameters for the 16 3D positions of the 4 tetrahedrons' LEDs. After these parameters were determined the robot presented the helmet to the cameras at a series of poses. At each pose, the LED tetrahedrons were illuminated one at a time. When either of the cameras could see a complete tetrahedron, Equation 3.29 was used to determine the position of the end effector (since this is the CF in which the LED positions were known) w.r.t. the observing camera. The camera's extrinsic parameters were then used to express this optical measurement of the end-effector pose in the reference CF. The difference between the optical measurement on the end effector pose and that provided by the robot arm itself was used as an accuracy measure for the helmet tracker. Table 6.6 provides the resulting accuracies per camera and for the system as a whole, for a test sequence of some 50 poses. The resulting system angular accuracy of 2.6 milliradians is comparable to systems [61–63] currently in operational use. This further demonstrates the calibration routines of Chapter 5 have definite real-world application.

# Chapter 7

# Simulation and calibration results

This chapter addresses the third of the three research questions posed in Section 1.2: *"Is a robotic arm based photogrammetric system sufficiently robust to measurement and movement noise?"* This chapter describes how the sensitivity analysis was performed for the APCCS. Specifically, the resultant stitching error (Section 6.1) caused by measurement errors in the calibration process is assessed. The stitching error was deemed to be the difference between the correct pixel on each camera's CCD that corresponds to a given point in space and the pixel position that was determined using the camera parameters as output by the APCCS. Since for physically realised systems, it is not possible to ever perfectly know the camera photogrammetric parameters, this sensitivity analysis is performed via simulation.

The sensitivity simulation is performed by synthesising noisy LED centroids in the format required by the calibration routines. The analysis simulates the APCCS robot arm moving through its specified sequence of points and adds errors with specified Probability Density Function (PDF) for each physical movement or measurement made by the APCCS to create noisy centroids and robot poses. A simplified version of this analysis was published in 2014 [8]. That simplified analysis did not cater for known pose errors of the robot arm or provide the mathematics of how the noisy LED centroids were synthesised.

The method used to synthesise the LED centroids is described in Section 7.1. This section includes the identification of the error sources and their contributions to both the noisy end effector pose and the noisy centroids is discussed.

Section 7.2 describes the design of the simulation experiment. The range of each of the noise parameters is presented as well are the combinations of the parameters to be tested. This section also includes the formal definition of the error metric used to assess how well a simulated calibration stitches images together.

Section 7.3 presents the results of the simulated experiment. The statistics of the results of the simulations are presented graphically and in tabular form and are discussed. This is followed by a detailed analysis of the correlation of the noise sources, individual calibration accuracies and stitching accuracies.

Section 7.4 places the results in context and presents the key findings of the simulation.

## 7.1 Creation of noisy LED centroids

This section describes how noisy centroids are synthesised in a manner relevant to the APCCS. Noisy LED centroids are simulated based on the physical set-up of a typical APCCS calibration apparatus. The information required to determine perfect LED centroids is:

1. The pose of the camera w.r.t. the robot CF (i.e. the camera's extrinsic parameters).
2. The pose of the end effector w.r.t. the robot CF.
3. Either the spatial position of the LED w.r.t. the end effector, or the relative poses of the PMJ mounting brackets, if the LED spatial offset is to be determined by the APCCS.
4. The camera's focal length, principal point, pixel size, resolution, and UD parameters (i.e. the camera intrinsic parameters).

The first three points above allow the position of the LED in the camera's CF to be determined. The last point then allows this 3D position to be converted into a 2D pixel position. In order to analyse the sensitivity of the algorithms to typical measurement errors the following four error sources have been identified:

1. The robot arm is unable to move to the exact required pose, but moves as close as it can to the requested pose and reports this resultant pose. Most of the APCCS routines use the reported robot arm pose, but the DU calibration (Section 5.5.1) assumes the points are along perfectly straight lines. This error was modelled using separate angular and spatial components (Equation 7.6).
2. The robot is unable to perfectly measure its resultant pose after each movement. This results in an additional, unknown deviation from the non-perfect pose described above. All the APCCS calibrations are affected by this error. This error was modelled with separate angular and spatial components (Equation 7.6).
3. The camera is not perfectly repeatably mountable on the kinematic mounting interfaces. This affects the calibrations of the focal length, both extrinsic poses, and the measurement of the LED spatial offset from the end effector. In contrast to the robot arm movement errors, which are assumed independent for each LED centroid, this error only changed when the camera was removed and placed on another mounting interface. This error was modelled with separate angular and spatial components (Equation 7.6).
4. The LED image centroid was not found with perfect precision. This error was a modelled as a spatial 2D error (Equation 7.6) in the pixel domain.

For a true reflection on the performance of a given APCCS, the PDF of the errors just described should be based on the characteristics of the APCCS components. These characteristics include physical effects such as thermal noise, pixel spatial sampling, fill factors, digital quantisation levels, and internal reflections in the lens. However the aim of this sensitivity study was not to determine the expected performance of the APCCS using a particular robot arm, kinematic mount, and camera. Rather the goal was to understand how the magnitudes of the errors affect the performance of the APCCS, which can aid the selection APCCS components. Therefore, the errors were modelled using a simple Gaussian PDFs.

Angular errors are modelled using the angle-axis representation (Section 3.3.3) of a 3D rotation. The axis is determined by generating a random UV whose end point is uniformly distributed (according to surface area) on a unit sphere [90]. The angle of rotation about this axis is a zero-mean Gaussian variable with a specified standard deviation. Equation 7.1 illustrates this:

$$\mathbf{R}^{\mathfrak{r},x} = \mathfrak{f}^{AAtoR}(\bar{U}^{\mathfrak{r},3}, \mathfrak{f}^{\mathrm{rn}}(x)) \tag{7.1}$$

where

$\mathbf{R}^{\mathfrak{r},x}$ = the desired random orientation matrix,

$\mathfrak{f}^{AAtoR}$ = as per Equation 3.12,

$\bar{U}^{\mathfrak{r},3}$ = a random UV (Equation 7.2), and

$\mathfrak{f}^{\mathrm{rn}}(x)$ = a zero mean Gaussian variable with standard deviation $x$.

3D spatial errors are simply modelled as scaled random UVs. The UV's end points are uniformly distributed on a unit sphere [90] and the magnitude is a zero-mean Gaussian with a specified standard deviation. Equation 7.2 shows how these vectors are calculated:

$$\bar{U}^{\mathfrak{r},3} = \begin{bmatrix} \cos(\theta_{P,\mathfrak{r}})\cos(\theta_{Y,\mathfrak{r}}) \\ \cos(\theta_{P,\mathfrak{r}})\sin(\theta_{Y,\mathfrak{r}}) \\ \sin(\theta_{P,\mathfrak{r}}) \end{bmatrix} \tag{7.2}$$

where

$\bar{U}^{\mathfrak{r},3}$ = desired random 3D UV,

$\theta_{P,\mathfrak{r}} = \cos^{-1}(\mathfrak{f}^{\mathrm{ru}}(-1,1))$,

$\theta_{Y,\mathfrak{r}} = \mathfrak{f}^{\mathrm{ru}}(-\pi,\pi)$,

$\mathfrak{f}^{\mathrm{ru}}(x,y)$ = uniformly distributed random variable between $x$ and $y$.

2D spatial errors are calculated in a manner identical to the 3D vectors except that the end point is on a unit circle rather than a unit sphere. Equation 7.3

provides the details:

$$\bar{U}^{\mathfrak{r},2} = \begin{bmatrix} \cos(\theta_{\mathfrak{r}}) \\ \sin(\theta_{\mathfrak{r}}) \end{bmatrix} \tag{7.3}$$

where

$\bar{U}^{\mathfrak{r},2}$ = desired random 2D UV,

$\theta_{\mathfrak{r}} = \mathfrak{f}^{\text{ru}}(-\pi, \pi)$, and

$\mathfrak{f}^{\text{ru}}(x, y)$ = uniformly distributed random variable between $x$ and $y$.

One of the requirements for generating noisy, distorted LED centroids is the ability to map undistorted pixel coordinates to distorted pixel coordinates. However, the Brown model (like all models) does not necessarily capture all the complexities observed in modelling DU and UD parameters. Therefore in order to simulate complexities beyond those that typical five to ten parameter Brown models can capture, a very high order DU model was used. A 17 parameter model was fitted to a dense $87 \times 60$ grid captured with a representative camera (the same used in Section 4.2). Rather than fitting a UD model, with the associated additional error, the DU model is used recursively with an optimisation algorithm such as Leapfrog (Section 3.2.2.4) to determine which distorted pixel position is associated with a given undistorted position. Equation 7.4 provides the mathematical details:

$$\begin{aligned} \bar{P}^d &= \mathfrak{f}^{\text{IRD}}(\bar{P}^u, \bar{DU}'_{\text{params}}) \\ &\equiv \underset{\bar{P}^d}{\text{argmin}} \left[ \|\bar{P}^u - \mathfrak{f}^{\text{Brown}}(\bar{P}^d, \bar{DU}'_{\text{params}})\|^2 \right] \end{aligned} \tag{7.4}$$

where

$\bar{P}^d$ = the desired distorted pixel position corresponding to $\bar{P}^u$,

$\bar{P}^u$ = the input undistorted pixel position,

$\mathfrak{f}^{\text{Brown}}$ = as per Equation 3.25 (Section 3.4.6 for parameter details), and

$\bar{DU}'_{\text{params}}$ = the camera's theoretical high order distorted to undistorted parameters.

The procedure to determine a set of noisy centroids is then the following:

1. Determine the noisy pose of the camera w.r.t. the robot CF by adding the camera w.r.t. the mount pose to the mount w.r.t. the robot pose and then adding an angular and spatial error of specified magnitudes as shown in Equation 7.5:

$$\begin{aligned}
\mathbf{R}^{n}_{cr} &= \mathbf{R}^{\mathfrak{r},\mathfrak{n}_{\mathfrak{ca}}}\mathbf{R}_{mr}\mathbf{R}_{cm}, \text{ and} \\
\bar{T}^{n}_{rcr} &= \bar{T}_{rmr} + \mathbf{R}_{mr}\bar{T}_{mcm} + \mathfrak{f}^{\mathrm{rn}}(\mathfrak{n}_{ct})\bar{U}^{\mathfrak{r},3} \qquad (7.5)
\end{aligned}$$

where

$$\begin{aligned}
\mathbf{R}^{n}_{cr}, \bar{T}^{n}_{rcr} &= \text{the noisy pose of the camera w.r.t. the robot CF,} \\
\mathbf{R}_{cm}, \bar{T}_{mcm} &= \text{the pose of the camera w.r.t. the mount,} \\
\mathbf{R}_{mr}, \bar{T}_{rmr} &= \text{the pose of the mount w.r.t. the robot CF,} \\
\mathfrak{r}, \mathfrak{n}_{\mathfrak{ca}} &= \text{standard deviation of camera angular error,} \\
\mathfrak{n}_{ct} &= \text{standard deviation of camera translation error,} \\
\mathbf{R}^{\mathfrak{r},\mathfrak{n}_{\mathfrak{ca}}} &= \text{random orientation matrix with axis evenly distributed on a} \\
&\quad\ \text{sphere and zero mean normally distributed angle with a} \\
&\quad\ \text{standard deviation of } n_{ca} \text{ as per Equation 7.1,} \\
\mathfrak{f}^{\mathrm{rn}}(x) &= \text{a zero mean Gaussian variable with standard deviation } x \text{, and} \\
\bar{U}^{\mathfrak{r},3} &= \text{3D UV evenly distributed on a sphere (Equation 7.2).}
\end{aligned}$$

2. Set the end effector to the ideal pose as requested by the APCCS.
3. Corrupt the ideal pose by an error with a specified PDF to simulate the robot arm not being able to move exactly to the required pose. This is the pose used for further calculations by the APCCS routines.
4. Further corrupt this pose by noise with a specified PDF to simulate the imperfect measurement by the robot arm of its own pose. This is the pose used in the rest of the noisy LED creation process.
5. Use the noisy end effector pose of Step 4 and the LED spatial offset to determine the spatial position of the LED in the robot's CF.
6. Use the camera extrinsic parameters to determine the LED translation w.r.t. the camera, in the camera's CF.

7. Use the camera's intrinsic parameters to project the LED onto the image plane and convert this to an undistorted pixel position.
8. Iteratively use the simulation's high-order ideal DU parameters to find the distorted pixel position that match the undistorted pixel position.
9. Corrupt this distorted pixel position by adding a 2D random error with a specified PDF and magnitude to get centroid's final position.
10. Save the noisy LED centre of step 9 and the noisy robot pose of step 3.
11. Go to step 2 and repeat for all required poses of the end effector.

This process is described mathematically in Equation 7.6:

$$\begin{aligned}
\bar{P}^{N}_{hv} &= \mathfrak{f}^{\text{LED\_Sim}}\left(\begin{array}{l} \mathbf{R}^{n}_{cr}, \bar{T}^{n}_{rcr}, \bar{UD}_{\text{params}}, \bar{I}_{\text{params}}, \mathbf{R}_{ar}, \bar{T}_{rar}, \\ \bar{T}_{ala}, \mathfrak{n}_{rtk}, \mathfrak{n}_{rtu}, \mathfrak{n}_{rak}, \mathfrak{n}_{rau}, \mathfrak{n}_{cpu} \end{array}\right) \qquad (7.6) \\
&\equiv \mathfrak{f}^{\text{IRD}}(\mathfrak{f}^{\text{P}\to\text{U}}(\bar{T}^{n}_{clc}, \bar{I}_{\text{params}}), \bar{DU}'_{\text{params}}) + \mathfrak{f}^{\text{rn}}(\mathfrak{n}_{cpu})\bar{U}^{\mathfrak{r},2}
\end{aligned}$$

where

$$\begin{aligned}
\mathfrak{f}^{\text{LED\_Sim}} &= \text{function to create a synthetic noisy LED centre,} \\
\mathfrak{f}^{\text{IRD}} &= \text{iterative reverse distortion as per Equation 7.4,} \\
\mathfrak{f}^{\text{P}\to\text{U}} &= \text{as per Equation 3.18,} \\
\bar{DU}'_{\text{params},j} &= \text{camera } j\text{'s theoretical high order DU parameters,} \\
\bar{I}_{params,j} &= \text{camera } j\text{'s intrinsic parameters,} \\
\mathfrak{f}^{\text{rn}}(x) &= \text{a zero mean Gaussian variable with standard deviation } x, \\
\bar{U}^{\mathfrak{r},2} &= \text{a random 2D UV (Equation 7.3),} \\
\bar{T}^{n}_{clc} &= \text{noisy position of LED w.r.t. camera,} \\
&= (\mathbf{R}^{n}_{cr})^{T}(\bar{T}^{n}_{rlr} - \bar{T}^{n}_{rcr}), \\
\bar{T}^{n}_{rlr} &= \bar{T}_{rar} + \mathfrak{f}^{\text{rn}}(\mathfrak{n}_{rtk})\bar{U}^{\mathfrak{r},3} + \mathfrak{f}^{\text{rn}}(\mathfrak{n}_{rtu})\bar{U}^{\mathfrak{r},3} + \mathbf{R}^{\mathfrak{r},\mathfrak{n}_{rau}}\mathbf{R}^{\mathfrak{r},\mathfrak{n}_{rak}}\mathbf{R}_{ar}\bar{T}_{ala}, \\
\bar{U}^{\mathfrak{r},3} &= \text{3D UV evenly distributed on a sphere (Equation 7.2),} \\
\mathbf{R}^{\mathfrak{r},\mathfrak{n}} &= \text{random rotation created by a Gaussian rotation around} \\
&\qquad \text{an evenly distributed random UV (Equation 7.1),} \\
\mathbf{R}^{n}_{cr}, \bar{T}^{n}_{rcr} &= \text{noisy pose of camera w.r.t. robot CF (Equation 7.5),} \\
\mathbf{R}_{ar}, \bar{T}_{rar} &= \text{pose of end effector w.r.t. the robot CF,}
\end{aligned}$$

$\bar{T}_{ala}$ = position of LED w.r.t. the end effector's CF,

$\mathfrak{n}_{rtk}$ = standard deviation of known robot translation error,

$\mathfrak{n}_{rtu}$ = standard deviation of unknown robot translation error,

$\mathfrak{n}_{rak}$ = standard deviation of known robot orientation error,

$\mathfrak{n}_{rau}$ = standard deviation of unknown robot orientation error, and

$\mathfrak{n}_{cpu}$ = standard deviation of unknown centroid pixel error.

## 7.2 Design of experiment

The physical apparatus simulated in this experiment is a simple two camera splayed array. The cameras are based on those used in the preliminary research (see Chapter 4) and have a 4.8 mm focal length lens with a $1600 \times 1200$ resolution and 5.5 $\mu$m pixel pitch. This gives an approximate HFOV of 80° per camera. Splaying the camera pair in azimuth by 60° results in an overlap of 20° and a combined HFOV of 140°. The multi-geometry stitching technique was used with a radius of 300 m and a horizontal stitch plane 20 m below the cameras. This simulates the cameras mounted on a building or ship.

While seemingly superficial this requires 66 independent parameters (33 per camera as per Table 6.1) to be determined. Assuming the LED spatial offset w.r.t. the robot end effector is unknown, Table 7.1 shows the robot movement sequences that are required for each simulation. The ranges for each error source's standard deviation for this evaluation are provided in Table 7.2. The range for each error was based on the published accuracies for several small robotic arms (Table 7.3) and kinematic mounting bases (Table 7.4).

All of the robot arms only had specifications for their reach and spatial accuracy. There was no indication of specify whether the accuracy value is the maximum error, 3 sigma error or the mean error. The published errors are also a combination of the known and unknown translation errors. It is apparent that there are two classes robots in Table 7.3 in terms of accuracy. The maximum standard deviation of 100 $\mu$m simulated for each of the known and unknown errors cater for

Table 7.1: Robot movement sequences required for two camera simulation.

| **Item** | **Calibration** | **Number of grids** | **Grid description** |
|---|---|---|---|
| Camera 1 | Total | 8 | - |
| | DU & UD calibrations | 1 | 3015 (67 × 45) points spanning 440 mm by 330 mm in 2D grid. |
| | Focal length | 2 | 2 observations of the same 80 point (20 tetrahedron) grid from 2 camera poses. |
| | Camera w.r.t. mount pose | 4 | Same 1323 (21 × 21 × 3) points spanning 200 mm by 200 mm by 100 mm in a 3D grid observed from 4 known mounting poses. |
| | Mount pose | 1 | 1323 (21 × 21 × 3) points spanning 500 mm by 300 mm by 200 mm in a 3D grid. |
| LED offset | | 4 | 2 cone and 2 axis movement grids: one each from 2 known mounting points. |
| Camera 2 | Total | 8 | - |
| | DU & UD calibrations | 1 | 3015 (67 × 45) points spanning 440 mm by 330 mm in 2D grid. |
| | Focal length | 2 | 2 observations of the same 80 point (20 tetrahedron) grid from 2 camera poses. |
| | Camera w.r.t. mount pose | 4 | Same 1323 (21 × 21 × 3) points spanning 200 mm by 200 mm by 100 mm in a 3D grid observed from 4 known mounting poses. |
| | Mount pose | 1 | 1323 (21 × 21 × 3) points spanning 500 mm by 300 mm by 200 mm in a 3D grid. |
| Complete simulation | | 20 | 8 per camera and 4 for the LED offset |

the more accurate class of robot arms. If the values are the RMS, or maximum 3 sigma error then a zero mean, 100 $\mu$m standard deviation PDF should exceed the expected range of accuracies of the robot arms.

Table 7.2: Simulation error ranges.

| **Error** | **Unit** | **Minimum** | **Maximum** |
|---|---|---|---|
| Known robot spatial | $\mu$m | 0 | 100 |
| Known robot angular | $\mu$rad | 0 | 100 |
| Unknown robot spatial | $\mu$m | 0 | 100 |
| Unknown robot angular | $\mu$rad | 0 | 100 |
| Camera mount spatial | $\mu$m | 0 | 200 |
| Camera mount angular | $\mu$rad | 0 | 100 |
| LED centroid error | pixels | 0 | 0.5 |

The angular accuracy values in Table 7.3 are calculated as the arctangent of the ratio of spatial error to the robot arm's reach. The error range of the simulation exceeds the range of the more accurate subset of robots.

Table 7.3: Comparison of robotic arms.

| **Robot** | | **Spatial accuracy ($\mu$m)** | **Reach (mm)** | **Angular accuracy ($\mu$rad)** | **Mass (kg)** |
|---|---|---|---|---|---|
| **Manufacturer** | **Model** | | | | |
| ABB | IRB120 | 10 | 580 | 17.2 | 25 |
| Denso | VS-050 | 20 | 520 | 38.5 | 29 |
| | VP-6242G | 20 | 432 | 46.3 | 15 |
| Fanuc | LRMate 200iD/4s | 20 | 36.4 | 172 | 25 |
| Kuka | LBR iiwa | 100 | 500 | 200 | 24 |
| | 6 R700 FIVVE | 30 | 706 | 42.5 | 50 |
| ST Robotics | R12-6 | 200 | 500 | 400 | 13 |
| | R17-6 | 200 | 750 | 267 | 21 |

Similar to the robot arms, the definition of the spatial accuracy for the kinematic bases is undefined. The accuracy range simulated far exceeds the value for the two mounts in Table 7.4 that have published spatial accuracies (particularly if they are the RMS errors). The simulated angular errors of the kinematic bases equals published accuracy of the Newport mount used in this work and exceeds the published values of the Eksmo and Thor Labs kinematic mounts in Table 7.4.

The dimensionality of the simulation can be either 4 or 7. The former is obtained by assuming the matching angular and spatial errors increase together and the

Table 7.4: Comparison of kinematic bases.

| **Kinematic base** | | **Repeatability** | |
|---|---|---|---|
| | | **Angular** | **Spatial** |
| **Manufacturer** | **Model** | **($\mu$rad)** | **($\mu$m)** |
| Eksmo | 850-0030 | 20 | 50 |
| Newport | M-BK-2A | 100 | |
| | M-BKL-4 | 100 | |
| Thor Labs | KB1x1 | 30 | 30 |
| | KB3x3 | 80 | 30 |

latter assumes that all the errors are independent. Initial testing and development was performed on a laptop with an Intel i7-740qm CPU. This CPU has 4 hyper-threaded cores, effectively providing 8 cores for processing. The simulation and calibration algorithms were implemented in a multi-threaded fashion to take maximal advantage of available processing power. On this development machine the best case simulation time observed was 3 minutes and the worst case was 12 minutes when using 3015 point DU and UD grids. Table 7.5 provides the estimated running times of the simulations, assuming that for statistical rigour at least 10 simulations were run for each error combination. Different values are provided for testing of each parameter independently of the others and for a full correlated scenario.

Table 7.5 clearly shows that for all but the most trivial correlated scenarios the duration quickly becomes unfeasible. This can be alleviated by using more powerful CPUs (the i7-740qm has a benchmark of 3241 [91] versus over 17000 for the latest CPUs [91]) or more than one computer. However, these are linear gains on an exponential problem. Therefore, a thorough independent analysis was performed. The independent analysis used 7 parameters with 6 steps for spatial noise sources, 5 steps for the angular sources and only every second point in the DU grid.

In order to quantify the resulting stitch accuracy of each simulation run, the stitches for the simulated and ideal calibrations are compared. For a set of points on the stitch within the valid azimuth and elevation ranges of the stitch, the corresponding pixel coordinates are calculated for each camera using both the

Table 7.5: Predicted simulation durations.

| **Para-meters** | **Steps** | **Combi-nations** | **Time**[†] | | |
|---|---|---|---|---|---|
| | | | **Best case** | **Worse case** | **Units** |
| 4, independent | 3 | 12 | 6.000 | 24.00 | hours |
| | 5 | 20 | 10.00 | 40.00 | hours |
| | 7 | 28 | 14.00 | 56.00 | hours |
| | 11 | 44 | 22.00 | 88.00 | hours |
| 7, independent | 3 | 21 | 10.50 | 42.00 | hours |
| | 5 | 35 | 17.50 | 70.00 | hours |
| | 7 | 49 | 24.50 | 98.00 | hours |
| | 11 | 77 | 37.50 | 154.0 | hours |
| 4, correlated | 3 | 64 | 32.00 | 128.0 | hours |
| | 5 | 1024 | 21.33 | 85.33 | days |
| | 7 | 16384 | 0.935 | 3.740 | years |
| | 11 | 4194304 | 2.394 | 9.576 | centuries |
| 7, correlated | 3 | 343 | 7.146 | 28.58 | days |
| | 5 | 16807 | 0.959 | 3.837 | years |
| | 7 | 823543 | 0.470 | 1.880 | centuries |
| | 11 | 1977326743 | 112.9 | 451.4 | millennia |

[†] using all 4 hyper-threaded cores of a single Intel i7-740qm CPU.

simulated and ideal calibration parameters. This allows the stitching accuracy to be assessed independently of the scene content, which affects the perceived stitching quality if objects in the scene are far from the stitch surface. The ideal calibration is known for the simulation as it is required to create the noisy centroids for the simulation (Section 7.1). The RMS magnitude of the differences between all the pairs of simulated and ideal pixel coordinates is calculated over the entire stitch. As the stitch is iterated over, only cameras which can see that stitch point contribute to the error. Equation 7.7 expresses this mathematically:

$$\mathrm{E}^{\mathrm{stitch}} = \sqrt{\frac{1}{w_{sum}} \left( \sum_{\alpha=\alpha_{min}}^{\alpha_{max}} \sum_{\beta=\beta_{min}}^{\beta_{max}} \sum_{i=0}^{N-1} \left( \|\bar{P}_i(\alpha,\beta) - \bar{P}'_i(\alpha,\beta)\|^2 w_{\alpha,\beta,i} \right) \right)} \tag{7.7}$$

where

$\mathrm{E}^{\mathrm{stitch}}$ = the simulation stitch error,

$$\begin{aligned}
w_{sum} &= \text{the sum of the binary weights,} \\
&= \sum_{\alpha=\alpha_{min}}^{\alpha_{max}} \sum_{\beta=\beta_{min}}^{\beta_{max}} \sum_{i=0}^{N-1} w_{\alpha,\beta,i}, \\
(\alpha, \beta) &= \text{azimuth and elevation of current stitch vector,} \\
[\alpha_{min}, \alpha_{max}] &= \text{the azimuth range of the stitch,} \\
[\beta_{min}, \beta_{max}] &= \text{the elevation range of the stitch,} \\
N &= \text{number of cameras in the stitch,} \\
\bar{P}_i(\alpha, \beta) &= \text{camera } i\text{'s image coordinate for the current stitch angles} \\
&\quad \text{using the simulated calibration (Equation 7.8),} \\
\bar{P}'_i(\alpha, \beta) &= \text{camera } i\text{'s image coordinate for the current stitch angles} \\
&\quad \text{using the ideal calibration (Equation 7.9),} \\
w_{\alpha,\beta,i} &= \text{binary weighting factor culling points outside camera } i\text{'s FOV,} \\
&= \begin{cases} 1 & \text{if } \bar{P}'_i(\alpha,\beta).h \in (0, R_{h,i}) \text{ and } \bar{P}'_i(\alpha,\beta).v \in (0, R_{v,i}) \\ 0 & \text{if otherwise} \end{cases}, \text{and} \\
(R_{h,i}, R_{v,i}) &= \text{the resolution of camera } i.
\end{aligned}$$

Equation 7.7 requires the pixel position that corresponds to a given camera's view of a point on the stitch surface as determined with the simulated camera calibrations. This is similar to the primary stitching algorithm given in Equation 6.1. The stitch point is expressed relative to the camera in question using the camera's total pose (i.e. the combination of camera w.r.t. the mount and the mount w.r.t. the reference poses). This point is then clipped to the image using the camera's focal length. The position of the vector in the plane is then converted to pixels using the camera's intrinsic parameters to yield the undistorted pixel position. The camera's UD parameters are then used to determine the distorted pixel position which is the desired pixel position of the camera corresponding to the stitch point. The exact formulation used for the simulation is given in Equation 7.8:

$$\bar{P}_i(\alpha,\beta) = \mathfrak{f}^{\text{Brown}}(\mathfrak{f}^{\text{P}\to\text{U}}(\mathbf{R}_{c_ir}^T\left(D^*\bar{U}_{rsr}(\alpha,\beta) - \bar{T}_{rc_ir}\right), \bar{I}^{\text{sim}}_{\text{params},i}), \bar{UD}^{\text{sim}}_{\text{params},i}) \tag{7.8}$$

where

$$\begin{aligned}
\bar{P}_i(\alpha,\beta) &= \text{image coordinate calculated from simulated calibration results} \\
&\qquad \text{at specified azimuth } \alpha \text{ and elevation } \beta, \\
\mathfrak{f}^{\text{Brown}} &= \text{as per Equation 3.25 (Section 3.4.6 for parameter details)}, \\
\mathfrak{f}^{\text{P}\to\text{U}} &= \text{as per Equation 3.18}, \\
\bar{UD}^{\text{sim}}_{\text{params},j} &= \text{camera } j\text{'s simulated UD parameters}, \\
\bar{I}^{\text{sim}}_{params,j} &= \text{camera } j\text{'s simulated intrinsic parameters}, \\
\bar{U}_{rsr}(\alpha,\beta) &= \begin{bmatrix} \cos(\beta)\cos(\alpha) \\ \cos(\beta)\sin(\alpha) \\ \sin(\beta) \end{bmatrix}, \\
D^*(\alpha,\beta) &= \min(D^s(\alpha,\beta), D^p(\alpha,\beta)), \\
D^s(\alpha,\beta) &= \text{distance from origin to stitch sphere, i.e. the stitch radius}, \\
D^p(\alpha,\beta) &= \text{distance from origin to stitch plane along } \bar{U}_{rsr}(\alpha,\beta), \\
&= \frac{D_{pr}}{\bar{U}_{rsr}(\alpha,\beta) \bullet \bar{U}_{rpr}}, \\
D_{pr} &= \text{distance of plane from stitch origin}, \\
\bar{U}_{rpr} &= \text{outward pointing normal UV of the plane, and} \\
\mathbf{R}_{c_ir}, \bar{T}_{rc_ir} &= \text{pose of camera } i \text{ resulting from simulated calibration.}
\end{aligned}$$

Equation 7.7 also requires the pixel position that corresponds to a given camera's view of a point on the stitch surface as determined by the ideal camera calibration. Unlike the normal stitching algorithm (Equation 6.1) only the DU lens parameters are known. Therefore numerical iteration is required to calculate the distorted pixel positions corresponding to the undistorted pixel position obtained from scaling the projection of the point on the stitch surface to the camera's inverted image plane. In all other respects this process is identical to that described for Equation 7.8. Equation 7.9 provides the mathematics for obtaining the ideal stitch pixel position:

$$\bar{P}_i'(\alpha,\beta) = \mathfrak{f}^{\mathrm{IRD}}(\mathfrak{f}^{\mathrm{P}\rightarrow\mathrm{U}}(\mathbf{R}_{c_ir}'^{T}\left(D^*\bar{U}_{rsr}(\alpha,\beta) - \bar{T}_{rc_ir}'\right), \bar{I}_{\mathrm{params},i}^{\mathrm{ideal}}), \bar{DU}_{\mathrm{params},i}^{\mathrm{ideal}}) \quad (7.9)$$

where

$$\begin{aligned}
\bar{P}_i'(\alpha,\beta) &= \text{ideal image coordinate for azimuth } \alpha \text{ and elevation } \beta,\\
\mathfrak{f}^{\mathrm{IRD}} &= \text{as per Equation 7.4 (Section 7.1 for parameter details)},\\
\mathfrak{f}^{\mathrm{P}\rightarrow\mathrm{U}} &= \text{as per Equation 3.18},\\
\bar{DU}_{\mathrm{params},j}^{\mathrm{ideal}} &= \text{camera } j\text{'s ideal DU parameters},\\
\bar{I}_{params,j}^{\mathrm{ideal}} &= \text{camera } j\text{'s ideal intrinsic parameters},\\
\bar{U}_{rsr}(\alpha,\beta) &= \begin{bmatrix}\cos(\beta)\cos(\alpha)\\ \cos(\beta)\sin(\alpha)\\ \sin(\beta)\end{bmatrix},\\
D^*(\alpha,\beta) &= \min(D^s(\alpha,\beta), D^p(\alpha,\beta)),\\
D^s(\alpha,\beta) &= \text{distance from origin to stitch sphere, i.e. the stitch radius},\\
D^p(\alpha,\beta) &= \text{distance from origin to stitch plane along } \bar{U}_{rsr}(\alpha,\beta),\\
&= \frac{D_{pr}}{\bar{U}_{rsr}(\alpha,\beta)\bullet\bar{U}_{rpr}},\\
D_{pr} &= \text{distance of plane from stitch origin},\\
\bar{U}_{rpr} &= \text{outward pointing normal UV of plane, and}\\
\mathbf{R}_{c_ir}', \bar{T}_{rc_ir}' &= \text{ideal pose of camera } i.
\end{aligned}$$

## 7.3 Results of experiment

This section discusses the results of the sensitivity study of the APCCS to the identified noise sources. In total, 33 different noise combinations were tested, with 10 simulations per sample to obtain an idea of the stochastic dependence of the stitch accuracy with regard to the noise sources. Each noise source was tested independently of the others. The magnitudes of the standard deviations that were tested can be seen in Tables 7.6 and 7.7 for the known and unknown robot pose error respectively. Table 7.8 shows the magnitudes that were evaluated for the camera mounting pose error and, finally, Table 7.9 shows the error standard deviations for the pixel evaluations.

Tables 7.6 to 7.9 also provide the minimum, mean, standard deviation and a histogram of the distribution of the samples. To further visualise the resultant stitch accuracy data, a box plot was created for each noise source/magnitude combination. Figure 7.1 shows these plots for the known and unknown spatial and angular errors made by the end effector in moving through its sequence of points. Figure 7.2 provides the plots for the mounting angular and spatial errors and the pixel centroid error. The box in each box plot extends from the 25th percentile to the 75th percentile and the middle line represents the median value. The whiskers extend to one standard deviation either side of the mean, which is plotted over the box plots using red squares.

Table 7.6: Known robot movement error effects on stitching accuracy.

| | **Known ang. noise** | | | | | **Known trans. noise** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **($\mu$Rad)** | | | | | **($\mu$m)** | | | | | |
| | **0.0** | **25.0** | **50.0** | **75.0** | **100.0** | **0** | **20** | **40** | **60** | **80** | **100** |
| Minimum | 0.99 | 0.58 | 6.22 | 0.81 | 0.61 | 0.99 | 0.76 | 0.56 | 0.70 | 0.58 | 0.89 |
| Mean | 9.30 | 10.90 | 15.26 | 7.18 | 10.74 | 9.30 | 11.28 | 8.18 | 11.38 | 9.68 | 9.25 |
| St. Dev. | 5.62 | 8.48 | 5.08 | 4.99 | 8.97 | 5.62 | 7.98 | 7.50 | 8.41 | 8.75 | 7.35 |
| $\mathfrak{E}^{\text{stitch}} < 1$ | 1 | 3 | 0 | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 1 |
| $1 \leq \mathfrak{E}^{\text{stitch}} < 3$ | 1 | 0 | 0 | 2 | 0 | 1 | 0 | 1 | 1 | 2 | 1 |
| $3 \leq \mathfrak{E}^{\text{stitch}} < 5$ | 1 | 0 | 0 | 1 | 3 | 1 | 3 | 2 | 1 | 2 | 1 |
| $5 \leq \mathfrak{E}^{\text{stitch}} < 10$ | 0 | 1 | 2 | 2 | 0 | 0 | 1 | 1 | 2 | 1 | 4 |
| $10 \leq \mathfrak{E}^{\text{stitch}} < 15$ | 6 | 2 | 2 | 3 | 0 | 6 | 2 | 2 | 1 | 0 | 0 |
| $15 \leq \mathfrak{E}^{\text{stitch}}$ | 1 | 4 | 6 | 1 | 5 | 1 | 3 | 2 | 4 | 4 | 3 |

Table 7.7: Unknown robot movement error effects on stitching accuracy.

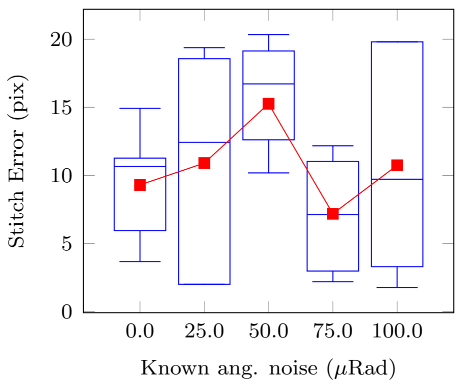
| | **Unknown ang. noise ($\mu$Rad)** | | | | | **Unknown trans. noise ($\mu$m)** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0.0** | **25.0** | **50.0** | **75.0** | **100.0** | **0** | **20** | **40** | **60** | **80** | **100** |
| Minimum | 0.99 | 0.59 | 0.59 | 0.63 | 0.56 | 0.99 | 0.63 | 0.77 | 0.57 | 0.97 | 0.72 |
| Mean | 9.30 | 7.28 | 10.37 | 7.81 | 8.95 | 9.30 | 8.88 | 8.97 | 11.81 | 8.20 | 12.67 |
| St. Dev. | 5.62 | 6.90 | 7.32 | 6.30 | 7.22 | 5.62 | 8.11 | 7.47 | 8.57 | 6.88 | 7.58 |
| $\mathfrak{E}^{\text{stitch}} < 1$ | 1 | 2 | 1 | 1 | 2 | 1 | 1 | 2 | 2 | 1 | 1 |
| $1 \leq \mathfrak{E}^{\text{stitch}} < 3$ | 1 | 1 | 2 | 3 | 0 | 1 | 2 | 1 | 1 | 1 | 0 |
| $3 \leq \mathfrak{E}^{\text{stitch}} < 5$ | 1 | 3 | 0 | 1 | 2 | 1 | 1 | 2 | 0 | 3 | 0 |
| $5 \leq \mathfrak{E}^{\text{stitch}} < 10$ | 0 | 1 | 3 | 0 | 3 | 0 | 3 | 1 | 1 | 1 | 3 |
| $10 \leq \mathfrak{E}^{\text{stitch}} < 15$ | 6 | 1 | 1 | 4 | 1 | 6 | 0 | 2 | 1 | 2 | 1 |
| $15 \leq \mathfrak{E}^{\text{stitch}}$ | 1 | 2 | 3 | 1 | 2 | 1 | 3 | 2 | 5 | 2 | 5 |

Table 7.8: Camera mounting pose uncertainty effect on stitching accuracy.

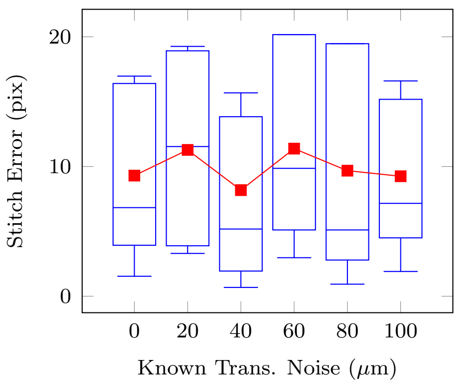
| | **Mount ang. noise ($\mu$Rad)** | | | | | **Mount trans. noise ($\mu$m)** | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **0.0** | **25.0** | **50.0** | **75.0** | **100.0** | **0** | **40** | **80** | **120** | **160** | **200** |
| Minimum | 0.99 | 0.70 | 0.65 | 1.42 | 0.68 | 0.99 | 0.60 | 0.57 | 0.61 | 0.75 | 0.57 |
| Mean | 9.30 | 9.28 | 8.39 | 11.93 | 13.21 | 9.30 | 6.06 | 8.20 | 6.64 | 12.76 | 7.89 |
| St. Dev. | 5.62 | 5.00 | 7.74 | 6.70 | 8.20 | 5.62 | 5.72 | 9.19 | 7.07 | 6.79 | 8.38 |
| $\mathfrak{E}^{\text{stitch}} < 1$ | 1 | 1 | 1 | 0 | 1 | 1 | 2 | 4 | 3 | 1 | 5 |
| $1 \leq \mathfrak{E}^{\text{stitch}} < 3$ | 1 | 0 | 2 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 |
| $3 \leq \mathfrak{E}^{\text{stitch}} < 5$ | 1 | 1 | 2 | 1 | 0 | 1 | 3 | 0 | 3 | 1 | 0 |
| $5 \leq \mathfrak{E}^{\text{stitch}} < 10$ | 0 | 3 | 2 | 2 | 2 | 0 | 2 | 1 | 2 | 1 | 1 |
| $10 \leq \mathfrak{E}^{\text{stitch}} < 15$ | 6 | 4 | 0 | 3 | 1 | 6 | 1 | 1 | 0 | 2 | 2 |
| $15 \leq \mathfrak{E}^{\text{stitch}}$ | 1 | 1 | 3 | 3 | 5 | 1 | 1 | 3 | 2 | 5 | 2 |

Table 7.9: Centroid error effects on stitching accuracy.

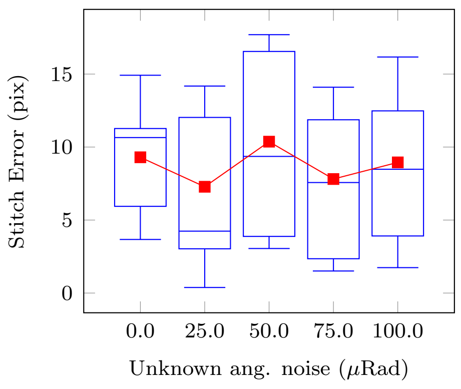
| | **Centroid noise (pix)** | | | | | |
|---|---|---|---|---|---|---|
| | **0.0** | **0.1** | **0.2** | **0.3** | **0.4** | **0.5** |
| Minimum | 0.99 | 0.56 | 1.04 | 0.99 | 0.96 | 1.05 |
| Mean | 9.30 | 8.25 | 12.71 | 13.25 | 11.14 | 7.92 |
| St. Dev. | 5.62 | 6.21 | 8.92 | 6.84 | 7.68 | 7.92 |
| $\mathfrak{E}^{\text{stitch}} < 1$ | 1 | 1 | 0 | 1 | 1 | 0 |
| $1 \leq \mathfrak{E}^{\text{stitch}} < 3$ | 1 | 1 | 2 | 1 | 2 | 3 |
| $3 \leq \mathfrak{E}^{\text{stitch}} < 5$ | 1 | 3 | 2 | 0 | 0 | 4 |
| $5 \leq \mathfrak{E}^{\text{stitch}} < 10$ | 0 | 0 | 0 | 0 | 1 | 0 |
| $10 \leq \mathfrak{E}^{\text{stitch}} < 15$ | 6 | 4 | 0 | 4 | 2 | 0 |
| $15 \leq \mathfrak{E}^{\text{stitch}}$ | 1 | 1 | 6 | 4 | 4 | 3 |



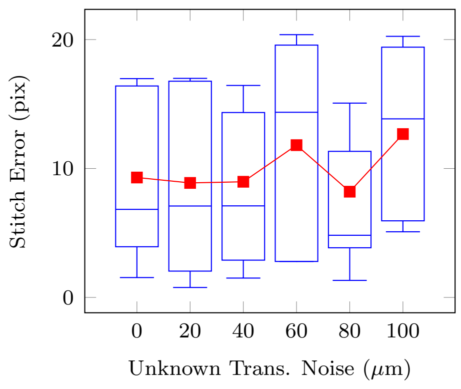
(a) Known angular error.



(b) Known translation error.

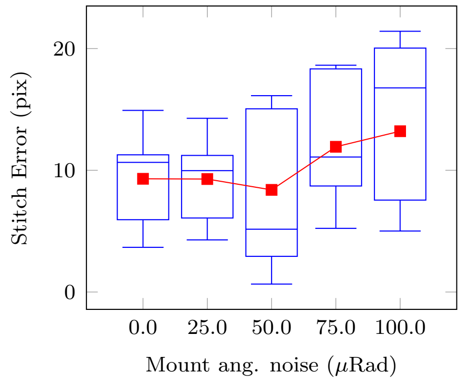


(c) Unknown angular error

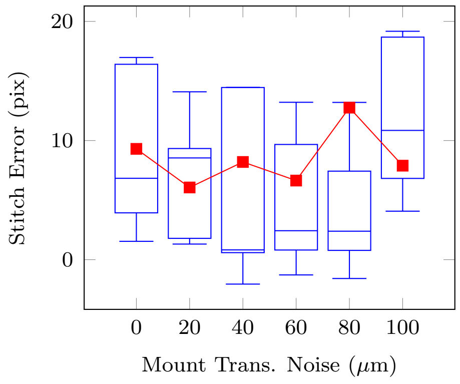


(d) Unknown translation error.

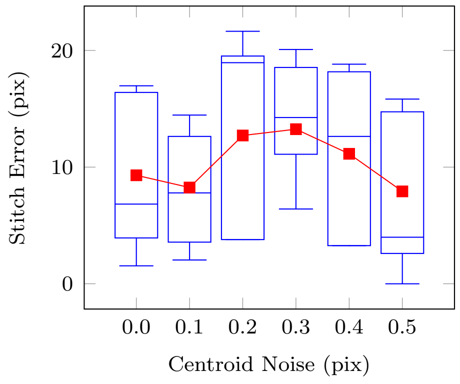
Figure 7.1: Stitching error due to robot movement errors.



(a) Mounting angular error.



(b) Mounting translation error.



(c) Pixel centroid error.

Figure 7.2: Stitching error due to mounting and pixel errors.

A review of Tables 7.6 to 7.9 and Figures 7.1 and 7.2 show that the effect of increasing the error in the ranges tested is minimal, with the exception of the mounting angular error which shows an upward trend. There are signs of a slight upward trend in both the mean and standard deviation for all the errors and slightly fewer resultant stitch accuracies of less than 3 pixels. It is worth recalling that the pixel error in this case is in the raw camera domain and not that of the panorama. This means that these errors may not be discernible in the output stitched panorama. It is also important to recall that the stitch accuracies use the calibrated parameters resulting from the processes described in Chapter 5. These processes make repeated use of nonlinear optimisation algorithms. Specifically, Leapfrog [26] was chosen to perform the optimisations because it is known to be more robust to noisy data and provide 'low local minima' [19]. The similarity of the stitch accuracy distributions over the noise magnitudes is thus not overly surprising.

What is not apparent from Tables 7.6 to 7.9 and Figures 7.1 and 7.2 is why many of the stitch accuracies had poor results of 10 pixels and higher. In all circumstances accuracies of around 20 pixels were obtained. For the $1600 \times 1200$, 4.8 mm focal length, and 5.5 $\mu$m pixel pitch cameras simulated, 20 pixels equates to approximately 1.3° of error. An analysis of the accuracies of each calibration performed in the simulations was undertaken. Both the returned minimised CFR as well as the error between the calibrated best value and the known correct value were assessed.

This yielded 16 values per simulation in addition to the seven noise levels and stitch accuracy for a total of 24 values. Table 7.10 lists and describes these 16 intermediate values in the order that their calibrations were performed.

Figures 7.3 and 7.4 show the global distributions of the parameters listed in Table 7.10 as calculated over the entire simulation. The global distribution of the stitching error is also shown. Their interpretation is the same as Figures 7.1 and 7.2. However, as disparate values are plotted alongside each other, no mean trend line is drawn and every plot has its units indicated.

The results of the 330 simulations were processed to determine the correlations

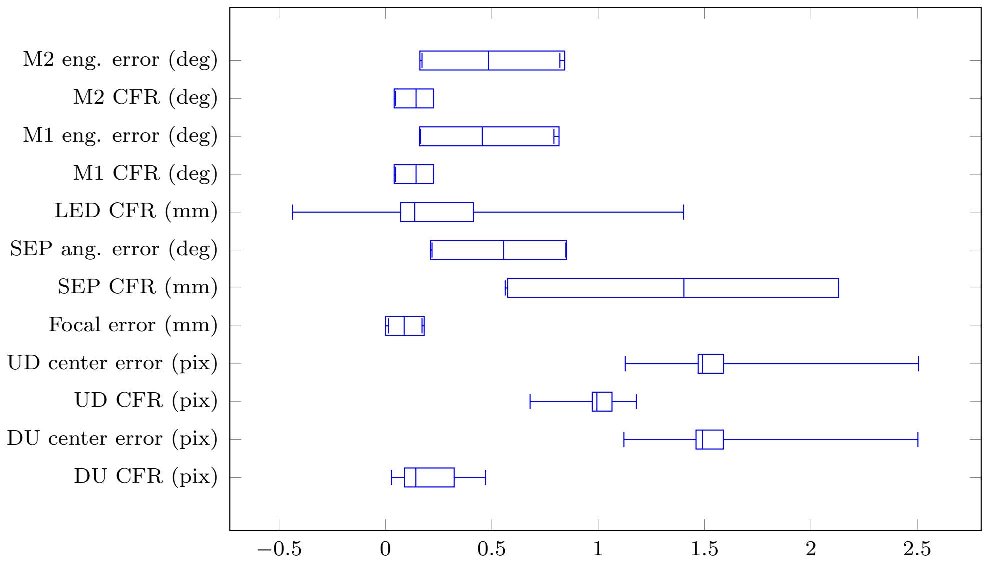


Figure 7.3: Resultant cost function and angular error distributions.

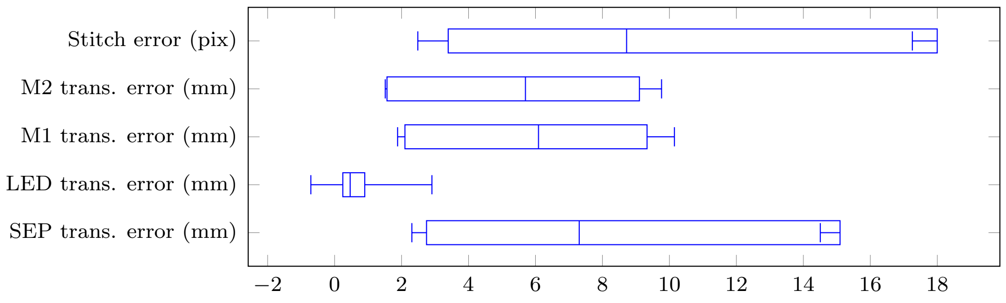


Figure 7.4: Resultant spatial and stitching error distributions.

Table 7.10: Simulation calibration accuracy measures.

| **Calibration** | **Section** | **Number of measures** | **Description of measures** |
|---|---|---|---|
| DU | 5.5.1 | 2 | CFR and the pixel distance between the determined and theoretical centres of distortion. |
| UD | 5.5.2 | 2 | CFR and the pixel distance between the determined and theoretical centres of distortion. |
| Focal length | 5.5.3 | 1 | Error between determined and theoretical focal lengths. |
| SEP | 5.5.5 | 3 | CFR and the angular and spatial errors between the determined and theoretical camera offsets. |
| LED spatial offset | 5.4.2 | 2 | CFR and the spatial error between the determined and theoretical spatial offsets. |
| Mount 1 pose | 5.5.4 | 3 | CFR and the angular and spatial errors between the determined and theoretical mount poses. |
| Mount 2 pose | 5.5.4 | 3 | CFR and the angular and spatial errors between the determined and theoretical mount poses. |

between the parameters listed in Table 7.10. The correlations of each parameter to the noise sources were only performed over the 50 or 60 values where that noise source was evaluated. This is because the noises sources were set to zero while the other sources were evaluated and would have strongly affected the distribution of the noise sources and skewed the correlation results. The correlations between the parameters are given in Table 7.11.

A complete cross correlation table is symmetrical around the main diagonal which has all unity entries (as these are the correlations of the parameters with themselves). To improve legibility the lower half of the table and the main diagonal were not populated. Additionally, since the seven noise parameters were evaluated independently the first seven columns are omitted. The calibration results in Table 7.11 are listed in the order they are performed. Thus for each parameter values to the right of the main diagonal are the ones that it affects. Values above the main diagonal are the effects of prior calibrations on that calibration.

Table 7.11 provides insight into the workings and co-dependencies of the APCCS calibrations. Both known and unknown robot angular noises values seem, in the ranges tested, to have little effect on the calibrations with the exception of the LED offset. This effect would likely be increased if the LED offset was larger. The spatial robot errors have a much larger effect, particularly on the DU calibration. The unknown translation error has a much larger effect the known error, particularly on the SEP calibration. Unlike the known error the unknown error has a small but noticeable effect on all the extrinsic calibrations.

The DU calibration is strongly dependent on the centroid error and to a lesser degree the robot translation errors. The centroid error also has the previously stated strong correlation with the DU calibration, as well as with LED spatial offset determination and a lesser effect on the SEP and extrinsic results. The mounting angular error has little effect on either of the distortion calibrations, which is expected, as the angle from which an LED grid is viewed does not alter the collinearity of the points in free space. It has a noticeable effect on the LED offset determination and all of the extrinsic calibrations. This result is expected as the SEP determination requires multiple mountings of the camera on kinematic bases, and the SEP error affects the mount pose determination, as discussed below.

The stitch accuracy is strongly dependent on the determination of the poses of the camera's mounting points with correlations ranging from 0.81 to 0.97. This is intuitive as any error in the camera pose immediately affects all the pixels for that camera. For example, if the camera is 3° off from its measured position all the calculations to create the stitch will be off by about 50 pixels. The stitch accuracy is more dependent on the angular component of the mount orientation than on the position, which is to be expected as the stitch surface is many orders of magnitude further away from the cameras than they are from each other. The only noise with which the stitch accuracy has any direct correlation is then, expectedly, the mounting angular error.

Table 7.11: Simulation parameter correlations.

| | **Parameter** | **8** | **9** | **10** | **11** | **12** | **13** | **14** | **15** | **16** | **17** | **18** | **19** | **20** | **21** | **22** | **23** | **24** |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | Known ang. noise | -0.03 | -0.09 | 0.02 | -0.09 | -0.04 | -0.00 | 0.08 | 0.09 | -0.53 | -0.37 | 0.03 | 0.03 | 0.10 | 0.03 | 0.06 | 0.10 | -0.02 |
| 2 | Known trans. noise | 0.26 | -0.06 | 0.04 | -0.06 | -0.01 | 0.03 | 0.04 | 0.04 | -0.14 | -0.12 | 0.00 | 0.02 | 0.05 | 0.00 | 0.03 | 0.05 | -0.01 |
| 3 | Unknown ang. noise | -0.04 | -0.04 | 0.03 | -0.04 | 0.00 | -0.01 | 0.01 | 0.02 | -0.52 | -0.38 | -0.00 | -0.00 | -0.02 | -0.00 | 0.00 | -0.02 | -0.00 |
| 4 | Unknown trans. noise | 0.50 | 0.16 | -0.16 | 0.16 | 0.10 | 0.24 | 0.22 | 0.14 | -0.15 | -0.03 | 0.21 | 0.18 | 0.17 | 0.21 | 0.18 | 0.15 | 0.12 |
| 5 | Mount ang. noise | -0.12 | -0.20 | 0.21 | -0.20 | 0.21 | 0.19 | 0.26 | 0.29 | 0.37 | 0.32 | 0.22 | 0.24 | 0.28 | 0.22 | 0.26 | 0.29 | 0.22 |
| 6 | Mount trans. noise | -0.03 | -0.05 | 0.06 | -0.05 | 0.08 | 0.07 | 0.08 | 0.08 | 0.12 | 0.05 | 0.07 | 0.10 | 0.08 | 0.07 | 0.10 | 0.09 | 0.08 |
| 7 | Centroid noise | 0.47 | 0.04 | -0.04 | 0.04 | -0.00 | 0.26 | 0.07 | 0.07 | 0.50 | 0.49 | 0.08 | 0.04 | 0.26 | 0.08 | 0.07 | 0.26 | 0.02 |
| 8 | DU CFR | - | 0.91 | -0.86 | 0.91 | -0.19 | 0.26 | 0.16 | -0.08 | -0.06 | 0.02 | 0.18 | 0.07 | 0.10 | 0.18 | 0.06 | 0.00 | -0.09 |
| 9 | DU center error | | - | -0.94 | 1.00 | -0.28 | 0.15 | 0.09 | -0.17 | -0.12 | -0.06 | 0.10 | -0.01 | 0.01 | 0.10 | -0.02 | -0.09 | -0.18 |
| 10 | UD CFR | | | - | -0.94 | 0.49 | 0.07 | 0.10 | 0.37 | 0.19 | 0.11 | 0.12 | 0.22 | 0.16 | 0.12 | 0.23 | 0.25 | 0.40 |
| 11 | UD center error | | | | - | -0.27 | 0.15 | 0.09 | -0.17 | -0.12 | -0.06 | 0.10 | -0.01 | 0.01 | 0.10 | -0.02 | -0.09 | -0.18 |
| 12 | Focal error | | | | | - | 0.88 | 0.86 | 0.93 | 0.31 | 0.27 | 0.90 | 0.94 | 0.76 | 0.90 | 0.92 | 0.77 | 0.99 |
| 13 | SEP CFR | | | | | | - | 0.96 | 0.92 | 0.31 | 0.31 | 0.99 | 0.97 | 0.87 | 0.99 | 0.97 | 0.83 | 0.93 |
| 14 | SEP ang. error | | | | | | | - | 0.96 | 0.28 | 0.29 | 0.98 | 0.98 | 0.92 | 0.98 | 0.99 | 0.89 | 0.91 |
| 15 | SEP trans. error | | | | | | | | - | 0.32 | 0.30 | 0.95 | 0.97 | 0.91 | 0.95 | 0.98 | 0.91 | 0.96 |
| 16 | LED CFR | | | | | | | | | - | 0.84 | 0.29 | 0.30 | 0.46 | 0.29 | 0.31 | 0.51 | 0.31 |
| 17 | LED trans. error | | | | | | | | | | - | 0.29 | 0.29 | 0.50 | 0.29 | 0.30 | 0.54 | 0.28 |
| 18 | M1 CFR | | | | | | | | | | | - | 0.99 | 0.88 | 1.00 | 0.99 | 0.85 | 0.95 |
| 19 | M1 ang. error | | | | | | | | | | | | - | 0.89 | 0.99 | 1.00 | 0.87 | 0.97 |
| 20 | M1 trans. error | | | | | | | | | | | | | - | 0.88 | 0.91 | 0.99 | 0.81 |
| 21 | M2 CFR | | | | | | | | | | | | | | - | 0.99 | 0.85 | 0.95 |
| 22 | M2 ang. error | | | | | | | | | | | | | | | - | 0.89 | 0.96 |
| 23 | M2 trans. error | | | | | | | | | | | | | | | | - | 0.81 |
| 24 | Stitch error | | | | | | | | | | | | | | | | | - |

It is unsurprising then that the stitch accuracy is also strongly correlated (0.83 to 0.99) with the SEP calibration accuracy. This is because the SEP accuracy directly affects the accuracy with which the camera mounting points are known. Indeed the correlation between the mounting bracket accuracies and the SEP calibration ranges are all above 0.97 for the mounting point orientations and above 0.83 for the mounting bracket translations.

The stitch accuracy, mount pose determination and the SEP are all strongly dependent on the focal length. The correlations for the mounting bracket spatial accuracies are 0.76 and 0.77. This dependence increases to approximately 0.90 for the angular components and 0.86 to 0.93 for the SEP calibration. The stitch accuracy has a correlation of 0.99 with the focal length! This is the core finding of the simulation: the sensitivity of not only the subsequent calibrations but of applications of the APCCS outputs are strongly linked to the focal length.

The focal length error only shows dependence on the UD calibration accuracy (correlation of 0.49) and slight dependence on the mounting angular error (0.21). This is due to this calibration requiring the camera to be remounted four times. The focal length's dependence on the DU calibration is -0.19. In contrast, the UD calibration is strongly linked to the DU calibration with a correlation of -0.86. This negative relationship is unsurprising as the same number of parameters are used for both the DU and UD calibrations. It has previously been shown [15, 92] that UD is more complex than DU and requires more parameters.

The UD calibration is seen to have a greater effect than the DU calibration. The UD calibration has a correlation of 0.40 with the stitch accuracy! The weak direct dependence of all the calibrations other than UD on the DU results is initially surprising until one considers the highly non-linear nature of all the calibrations. Each calibration does show strong dependence on the calibration performed immediately prior to it. It is also worth noting that the majority of resultant DU errors are better than a half a pixel RMS over the camera's FOVs, as evidenced by Figure 7.3. It is possible that at higher levels of residual distortion error stronger relationships will become apparent.

## 7.4 Simulation conclusion

The resultant stitch accuracies occupy a wide range of values from 0 to 20 pixels of error. Figures 7.3 and 7.4 show this deviation exists within each noise band and no strong trends emerge as noise levels are increased. Table 7.11 shows that stitch accuracy is most dependent on the UD calibration and focal error (which affects subsequent calibrations). Neither the focal error nor the stitch accuracy have any strong dependence on any of the noise sources, except for the angular mounting accuracy. The variation in accuracies is present even in the zero-noise case which may be indicative that the globally optimal calibration solution is not always being found. This may be due to a combination of the cost functions being too insensitive, or due to the probabilistic nature of global optimisations employed. Future research may address these concerns.

In conclusion, the posed research question of the calibration robustness with respect to noise can be answered. Tables 7.6 through 7.9 show that for all noise sources and magnitudes tested, the best accuracies achieved always fall in the range of 0.56 to 1.42 pixels of error and the stitch error distributions do not change significantly. It can thus be stated that the system is indeed robust to noise.

# Chapter 8

# Conclusions

This chapter places the work done in context. Firstly, a summary of the results are provided in Section 8.1. The original research questions are revisited and addressed in Section 8.2. Thereafter, key findings of the work are presented in Section 8.3. Finally suggestions for future work are presented in Section 8.4.

## 8.1 Results Summary

This work focuses on the intrinsic and extrinsic photogrammetric calibration of cameras and the use thereof for real-world applications. A summary of the methods employed can be found in Section 1.4. This section focusses on the results and research decisions that stemmed from each chapter.

In Chapter 1 the need for an adaptable versatile multi-spectral camera calibration equipment is identified. The decision to base the APCCS on a robotic arm is taken, with the proviso that the APCCS must have an extended range of capabilities compared to the other automated calibration systems discussed.

Chapter 2 provides a review of the relevant literature regarding numerical optimisation, camera calibration, camera based pose estimation, and photogrammetric stitching. As the calibration is not required to be real time, robust algorithms which can be used for black-box functions are selected.

The mathematical building bocks on which the rest of the research are based are presented in Chapter 3. The details of the chosen global and local optimisation routines are presented as are the fundamental image processing, trigonometric and photogrammetric algorithms that are subsequently used.

Chapter 4 provides the first research results from the APCCS development. The goal of the chapter is to determine which distortion modelling methods and which calibration targets and associated image processing routines provided the best results. Section 4.1 investigates which distortion modelling method provides the best characterisation. The author's previous distortion modelling work [16, 17] using high order Brown models, which already provided leading results, is used as a benchmark. It is seen after a feasibility study (Section 4.1.1) and subsequent optimisation study (Section 4.1.2) that ANNs could be trained successfully to characterise distortion. The final ANNs produced results superior to all published results with the exception of the benchmark. Two papers were published on the ANN feasibility study [10] and the improved ANN modelling [11] respectively.

The analysis of calibration targets and image processing routines is presented in Section 4.2. Circular and square calibration targets are presented together with two image processing methods each. The results are processed using high order Brown models [16, 17] and the popular OpenCV toolkit [4]. These results are benchmarked using by triangulating real-world 3D points from images in a common dataset. It is seen that high order Brown models consistently outperform OpenCV. Despite square patterns having better triangulation results overall, it was decided to use circular patterns with ellipse fitting as these patterns are more representative of an LED or other light source mounted on the robot arm. A paper [5] was published on an expanded version of this study which included linear calibration targets.

Section 5 contains all the algorithms developed for camera calibration with the APCCS. Both the routines to calibrate the APCSS itself, as well as the routines to calibrate cameras with the APCCS are presented in detail. Theses routines satisfy all the requirements of an ideal calibration system as listed in Section

1.1.3. This resulted in the APCCS being internationally patented [9] and a paper [7] being presented on it. The novelty of the APCCS is both in its unique combination of abilities and in several of the algorithms developed for calibration. The method to measure the spatial offset of the LED w.r.t. the end effector and the tetrahedral based determination of the focal length were deemed novel.

The assessment of the real-world applicability of the APCCS is dealt with in Chapter 6. Section 6.1 looks at photogrammetric stitching using cameras calibrated with the APCCS. Sections 6.1.1 and 6.1.2 present the work on the improved stitching algorithms and novel real-time blending algorithms respectively. These sections resulted in one paper on the registration [12] and two papers on the blending [13, 14]. Thereafter, both subjective stitching results and quantified errors based on photogrammetrically stitching image points found with feature based methods are presented. Results for two multispectral systems show that the stitching is accurate despite using cameras of different fields of view, spectra, and resolutions. This shows that the APCCS is both versatile and applicable to real-world applications.

Section 6.2 provides an overview of the results obtained for optical helmet tracking using cameras calibrated by the APCCS. The results obtained with a simple laboratory prototype tracker provide helmet pose estimation with an accuracy comparable to operational systems. This section provides further evidence of the practical applicability of the APCCS. An overview of the APCCS and its use for stitching and tracking was published [7].

Chapter 7 investigates the robustness of the APCCS to expected noises sources in the calibration process. The identification and synthesis of artificially noise corrupted data is presented in Section 7.1. Section 7.2 describes the design of the experiment, including error ranges and the stitching based error metric. The results of the simulation are presented and discussed in detail in Section 7.3. It is seen that the spread of stitching accuracies does not vary considerably over the noise ranges tested, thus showing the APCCS outputs are robust to noise. A deeper analysis of the correlations of the stitching accuracies, APCCS calibration outputs, and noise levels is also presented. In this analysis it is seen that the focal length calibration and camera angular mounting repeatability are critical as they have a strong influence on the extrinsic parameter calibrations which then directly degrade the stitching accuracy. A simplified version of this analysis [8] has been published.

## 8.2 Research question assessment

This section looks at each of the original research questions posed in Section 1.2 and answers them in light of the work done.

### 8.2.1 Research question 1

Can an automated photogrammetric camera calibration system meeting the criteria listed in Section 1.1.3 be created, using a robotic arm?

Yes. Chapter 5 provided the mathematical calibration routines. These routines were able to determine the DU and UD lens characterisations, optimal focal length and the pose of the camera. The pose was further split into two components to aid deployments. Additional algorithms were derived to be able to determine the offset of a light source from the end effector to allow light sources to be swapped to calibrate cameras of different spectra. None of the calibrations require knowledge of the pose of the mounting point relative to the robot arm, thus allowing the mount to be moved to facilitate the calibration of cameras with different FOVs. The number of positions in a robot movement sequence is configurable in order to trade off calibration time versus calibration accuracy. Multiple resolutions are supported by capturing the light sources' image coordinates one point at a time. The sequential capture allows small deltas in position which would otherwise cause the light sources to overlap in the camera images.

Chapter 6 provides proof, in Section 6.1, that the above is true by presenting stitch results for cameras of different resolutions, spectra, FOVs, and arranged in both linear and 2D arrays. Additional proof is briefly presented in the context of helmet tracking in Section 6.2, where cameras of different resolutions and FOVs are used. The helmet tracking application provided pose measurements in millimeters and degrees thus showing that the APCCS calibration can be used for absolute and not merely relative measurements.

### 8.2.2 Research question 2

Are the calibration parameters produced by such a system suitable for real world applications?

Yes. Chapter 6 demonstrates in Section 6.1 that multi-spectral photogrammetric stitching of cameras with different resolutions, spectra, and FOVs is possible with the outputs. This is verified using two different physical systems and real recorded data. Further evidence is proved in Section 6.2 where cameras of different resolutions and multiple FOVs are used to create a laboratory helmet tracker prototype.

### 8.2.3 Research question 3

Is a robotic arm based photogrammetric system sufficiently robust to measurement and movement noise?

Yes. Chapter 7 modelled the expected primary sources of noise in the calibration APCCS and used these values to create realistic synthetic data for the calibration routines of Chapter 5. Multiple simulations were performed for different magnitudes of each noise source while the resultant calibration and photogrammetric stitch accuracies were recorded. It was seen that while the spread of the resultant stitch accuracies is wider than desired, the spread is largely independent of stitch noise in the ranges tested. This shows that the system is indeed robust to noise.

## 8.3 Findings

The following are results from the work done:

1. The APCCS is able to accurately determine all the photogrammetric parameters of cameras of multiple FOVs, spectra, and resolutions.
2. The APCCS calibration outputs were successfully used to photogrammetrically stitch multi-spectral panoramas. Visual analysis showed the stitch to be correct. Qualitative analysis, based on the projection of matched SIFT and SURF features in overlapping regions of adjacent cameras, found the stitch to be accurate to better than $0.3°$.
3. The APCCS calibration outputs were successfully used to create a laboratory helmet tracker prototype which yielded accuracies comparable to existing deployed systems.
4. The APCCS is found to be resilient to noise in the robot movement, camera mounting and light source image localisation.
5. It is shown that explicit polar based models for lens distortion marginally outperformed ANN based black box modelling of lens distortion.
6. The ANN based lens inverse distortion (i.e. UD) modelling outperforms all the available models in literature except the author's prior work on precision calibration [16, 17].
7. The lens distortion modelling in this work significantly outperforms that of OpenCV (the de facto standard) for monocular localisation and 3D measurements.
8. Photogrammetric stitching, while sensitive to depth, is significantly more resilient than feature based stitching for uncontrolled outdoor scenes.
9. The focal length characterisation is a key calibration which directly affects the subsequent extrinsic calibrations and resultant stitching accuracy performed using the calibrated parameters.
10. The camera angular mounting repeatability was found to be the noise source with the greatest affect on calibration and stitching accuracy.

## 8.4 Future research

The following is suggested to continue this research:

1. Development of an all optical method to calibrate the PMJ, thereby allowing the APCCS to perform a complete self calibration.
2. Determine the principal point in conjunction with the focal length and use this value for image to vector transformations (and vice versa) rather than the DU and UD centres.
3. Find a better method to determine the similarity of two poses, this will improve the focal length calibration.
4. Investigate optimal 3D movement sequences of the robot arm for each calibration.
5. Investigate calibration assessment routines to determine if reprocessing the data may yield better calibration and stitching results.
6. Investigate better combinations of calibration cost functions and optimisation routines to yield a narrower spread of resultant calibration and stitching accuracies.
7. Investigate real-time image equalisation algorithms to mitigate the effects of uncontrolled lighting conditions on the camera images. This will improve the performance of image feature detection, description and matching algorithms, and result in more aesthetically pleasing stitched panoramas.
8. Investigate exploiting the depth sensitivity of the stitching algorithm by using a plane sweep to perform stereo passive ranging.

# Bibliography

[1] C. J. Willers, *Electro-Optical System Analysis and Design: A Radiometry Perspective*, vol. PM236. Bellingham, WA: SPIE Press, 2013.

[2] C. Murcott, F. D. Plessis, and J. Meyer, "A critique on previous work in vision aided navigation," in *AFRICON, 2011*, Sept 2011.

[3] K. Smith, A. Corregedor, C. Murcott, B. Andrews, S. Holte, M. Furrutter, M. Evans, J. Carroll, F. du Plessis, and J. Meyer, "Design and implementation of an autonomous hybrid vehicle," in *AFRICON, 2011*, Sept 2011.

[4] G. Bradski, "The OpenCV library," *Dr. Dobb's Journal of Software Tools*, 2000.

[5] J. P. de Villiers, F. J. Wilson, and F. C. Nicolls, "The effect of lens distortion calibration patterns on the accuracy of monocular 3D measurements," in *Proceedings of the 22nd Annual Symposium of the Pattern Recognition Association of South Africa*, vol. 1 of *PRASA 2011*, pp. 1–6, 2011.

[6] R. A. Peters, S. Atkins, D. J. Kim, and A. Nawab, "System and method for automatic calibration for stereo images," 3 2011. Patent Number US 2011/0063417 A1.

[7] J. P. de Villiers, R. S. Jermy, and F. C. Nicolls, "A versatile photogrammetric camera automatic calibration suite for multispectral fusion and optical helmet tracking," in *SPIE Defense, Security and Sensing*, vol. 90860W, pp. 1–9, 2014.

[8] J. P. de Villiers and F. C. Nicolls, “A study on the sensitivity of photogrammetric camera calibration and stitching,” in *Proceedings of the 25th Annual Symposium of the Pattern Recognition Association of South Africa*, vol. 1 of *PRASA 2014*, 2014.

[9] J. P. de Villiers and J. Cronje, “A method of calibrating a camera and a system therefor,” 11 2012. Patent Number WO 2014083386 A2.

[10] J. P. de Villiers and F. C. Nicolls, “Application of neural networks to inverse lens distortion modelling,” in *Proceedings of the 21st Annual Symposium of the Pattern Recognition Association of South Africa*, vol. 1 of *PRASA 2010*, pp. 63–68, 2010.

[11] J. P. de Villiers, J. Cronje, and F. C. Nicolls, “Improved neural network modeling of inverse lens distortion,” in *SPIE Defense, Security and Sensing*, vol. 8056, pp. 80560L–80560L–9, 2011.

[12] J. P. de Villiers and J. Cronje, “Improved real-time photogrammetric stitching,” in *Proc. SPIE 8744, Automatic Target Recognition XXIII*, vol. 8744, pp. 874406–874406–9, 2013.

[13] J. Cronje and J. P. de Villiers, “A comparison of image features for registering LWIR and visual images,” in *Proceedings of the 23nd Annual Symposium of the Pattern Recognition Association of South Africa*, vol. 1 of *PRASA 2012*, 2012.

[14] J. P. de Villiers and R. S. Jermy, “An analysis of fusion algorithms for LWIR and visual images,” in *Proceedings of the 24th Annual Symposium of the Pattern Recognition Association of South Africa*, vol. 1 of *PRASA 2013*, 2013.

[15] J. P. de Villiers, *Correction of radially asymmetric lens distortion with a closed form solution and inverse function, (Master’s thesis)*. Pretoria, RSA: University of Pretoria, 2008.

[16] J. P. de Villiers, F. W. Leuschner, and R. Geldenhuys, “Centi-pixel accurate real-time inverse distortion correction,” in *Proceedings of the 2008 Inter-*

*national Symposium on Optomechatronic Technologies*, vol. 7266 of *ISOT 2008*, pp. 1–8, 2008.

[17] J. P. de Villiers, F. Leuschner, and R. Geldenhuys, "Modeling of radial asymmetry in lens distortion facilitated by modern optimization techniques," in *SPIE Electronic Imaging*, vol. 7539, p. 75390J, SPIE, 2010.

[18] J. P. de Villiers, "Real-time stitching of high resolution video on COTS hardware," in *Proceedings of the 2009 International Symposium on Optomechatronic Technologies*, vol. 9 of *ISOT 2009*, pp. 46–51, 2009.

[19] J. A. Snyman, *Practical Mathematical Optimization.* New York, USA: Springer, 2005.

[20] R. L. Burden and J. D. Faires, *Numerical Analysis.* Pacific Grove, USA: Brookes Cole, 2001 (seventh edition).

[21] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *Computer Journal*, vol. 7, pp. 140–054, 1964.

[22] E. Polak and G. Ribière, "Note sur la convergence de mèthodes de directions conjugues," *ESAIM: Mathematical Modelling and Numerical Analysis - Modlisation Mathmatique et Analyse Numrique*, vol. 3, no. R1, pp. 35–43, 1969.

[23] K. Levenberg, "A method for the solution of certain non-linear problems in least squares," *Quarterly Applied Mathematics*, vol. 2, pp. 164–168, 1944.

[24] D. W. Marquardt, "An algorithm for least-squares estimation of nonlinear parameters," *Journal of the Society for Industrial Applied Mathematics*, vol. 2, pp. 431–441, 1963.

[25] MATLAB, *version 7.10.0 (R2010a).* Natick, Massachusetts: The MathWorks Inc., 2010.

[26] J. A. Snyman, "An improved version of the original leap-frog dynamic method for unconstrained minimization: LFOP1(b)," *Applied Mathematics and Modelling*, vol. 7, pp. 216–218, 1983.

[27] A. Neumaier, "Complete search in continuous global optimization and constraint satisfaction," *Acta Numerica*, vol. 13, pp. 271–369, 2004.

[28] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. pp. 671–680, 1983.

[29] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, 1995.

[30] R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, Dec. 1997.

[31] J. Holland, "Genetic algorithms and the optimal allocation of trials," *SIAM Journal on Computing*, vol. 2, no. 2, pp. 88–105, 1973.

[32] M. Srinivas and L. Patnaik, "Genetic algorithms: a survey," *Computer*, vol. 27, pp. 17–26, June 1994.

[33] A. E. Conrady, "Decentered lens systems," *Monthly Notices of the Royal Astronomical Society*, vol. 79, pp. 384–390, 1919.

[34] D. C. Brown, "Decentering distortion of lenses," *Photogrammetric Engineering*, vol. 7, pp. 444–462, 1966.

[35] D. C. Brown, "Close range camera calibration," *Photogrammetric Engineering*, vol. 8, pp. 855–855, 1971.

[36] T. A. Clarke and J. G. Fryer, "The development of camera calibration methods and models," *The Photogrammetric Record*, vol. 16, no. 91, pp. 51–66, 1998.

[37] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses," *IEEE Journal of Robotics and Automation*, vol. 3, pp. 323–344, 1987.

[38] Z. Zhang, "Flexible camera calibration by viewing a plane from unknown orientations," in *Proceedings of the 1999 Conference on Computer Vision and Pattern Recognition*, vol. 1 of *CVPR 99*, pp. 666–673, 1999.

[39] G. Stein, "Analytically solving radial distortion parameters," in *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, vol. 1 of *CVPR 97*, pp. 602–608, 1997.

[40] J. Bouguet, "Camera calibration toolbox for Matlab." http://www.vision.caltech.edu/bouguetj/calib_doc/.

[41] R. Sagawa, M. Takatsuji, T. Echigo, and Y. Yagi, "Calibration of lens distortion by structured-light scanning," in *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, pp. 832–837, Aug 2005.

[42] D. Claus and A. W. Fitzgibbon, "A rational function lens distortion model for general cameras," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 213–219, 2005.

[43] F. M. Candocia, "A scale-preserving lens distortion model and its application to image registration," in *Proceedings of the 2006 Florida Conference in Recent Advances in Robotics*, vol. 1 of *FCRAR 2006*, pp. 1–6, 2006.

[44] J. Mallon and P. F. Whelan, "Precise radial un-distortion of images," in *Proceedings of the 17th International Conference on Pattern Recognition*, vol. 1 of *ICPR 2004*, pp. 18–21, 2004.

[45] J. Harguess and S. Strange, "Infrared stereo calibration for unmanned ground vehicle navigation," in *Unmanned Systems Technology XVI,*, vol. 90840S, pp. 1–8, 2014.

[46] L. Lucchese and S. K. Mira, "Using saddle points for subpixel feature detection in camera calibration targets," in *Proceedings of the Asia-Pacific Conference on Circuits and Systems*, vol. 2, pp. 191–195, 2002.

[47] D. Chen and G. Zhang, "A new sub-pixel detector for X-corners in camera calibration targets," in *WSCG (Short Papers)*, pp. 97–100, 2005.

[48] J. I. Jeong, S. Y. Moon, S. G. Cho, and D. Rho, "A study on the flexible camera calibration method using a grid type frame with different line widths," in *Proceedings of the 41st SICE Annual Conference*, vol. 2, pp. 1319–1324, 2002.

[49] O. Silven and J. Heikkila, "Calibration procedure for short focal length off-the-shelf CCD cameras," in *Proceedings of the 13th International Conference on Pattern Recognition*, vol. 1, pp. 166–170, 1996.

[50] W. Yu, "An embedded camera lens distortion correction method for mobile computing applications," *IEEE Transactions on Consumer Electronics*, vol. 49, no. 4, pp. 894–901, 2003.

[51] A. Redert, E. Hendriks, and J. Biemond, "Accurate and robust marker localization algorithm for camera calibration," in *3D Data Processing Visualization and Transmission, 2002. Proceedings. First International Symposium on*, pp. 522–525, 2002.

[52] A. G. J. Nijmeijer, M. A. Boer, C. H. Slump, M. M. Samson, M. J. Bentum, G. J. Laanstra, H. Snijders, J. Smit, and O. E. Herrmann, "Correction of lens-distortion for real-time image processing systems," in *VLSI Signal Processing, VI, 1993., [Workshop on]*, pp. 316–324, Oct 1993.

[53] I. Tsatsakis, E. Kayafas, V. Loumos, and G. Cambourakis, "Using low cost video cameras in automation: a close range photogrammetry approach," in *Industrial Electronics, 1995. ISIE '95., Proceedings of the IEEE International Symposium on*, vol. 2, pp. 523–528, Jul 1995.

[54] W. Zheng, Y. Shishikui, Y. Kanatsugu, Y. Tanaka, and I. Yuyama, "A high-precision camera operation parameter measurement system and its application to image motion inferring," *IEEE Transactions on Broadcasting*, vol. 47, no. 1, pp. 46–55, 2001.

[55] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, pp. 381–395, Jun 1981.

[56] L. Kneip, D. Scaramuzza, and R. Siegwart, "A novel parametrization of the perspective-three-point problem for a direct computation of absolute camera position and orientation," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pp. 2969 –2976, june 2011.

[57] A. Katake and H. Choi, "VisNAV 100: a robust, compact imaging sensor for enabling autonomous air-to-air refueling of aircraft and unmanned aerial vehicles," in *Image Processing: Machine Vision Applications III*, vol. 7538 of *IE2010*, pp. 1–11, SPIE, 2010.

[58] A. Katake, C. Bruccoleri, P. Singla, and J. L. Junkins, "Landingnav: a precision autonomous landing sensor for robotic platforms on planetary bodies," in *Intelligent Robots and Computer Vision XXVII: Algorithms and Techniques*, vol. 7539 of *IE2010*, pp. 1–12, SPIE, 2010.

[59] J. Stavnitzky and F. Rivollier, "3D vision for nuclear reactor retrofit tool docking," in *Image Processing: Machine Vision Applications III*, vol. 7538 of *IE2010*, pp. 1–12, SPIE, 2010.

[60] R. Atac and E. Foxlin, "Scorpion hybrid optical based inertial tracker (HObIT)," in *Head and Helmet Mounted Displays XVIII: Design and Applications*, vol. 873502 of *SPIE*, 2013.

[61] R. Atac, S. Spink, T. Calloway, and E. Foxlin, "Scorpion hybrid optical based inertial tracker (HObIT) test results," in *Head and Helmet Mounted Displays XIX: Design and Applications*, vol. 873502 of *SPIE Defense Peace Safety and Security*, 2014.

[62] J. Larsson and T. Blomqvist, "The Cobra helmet mounted display system for Gripen," in *Head and Helmet Mounted Displays XIII: Design and Applications*, vol. 695505 of *SPIE Defense Peace Safety and Security*, 2013.

[63] A. Cameron, "Heads up and eyes out, advances in head mounted displays capabilities," in *Display Technologies and Applications for Defense, Security and Avionics VII*, vol. 87360G of *SPIE Defense Peace Safety and Security*, 2013.

[64] J. P. de Villiers and F. P. J. le Roux, "Omnidirectional maritime surveillance," in *Proceedings of the 3rd CSIR Biennial Conference: Science Real and Relevant*, 2010.

[65] S. Aburmad, "Panoramic thermal imaging: Challenges and tradeoffs," in *Infrared Technologies and Applications XL*, vol. 9070-11 of *SPIE Defense Peace Safety and Security*, 2014.

[66] Immersive Media, "360 video." http://www.immersivemedia.com/tag/360-video, 2014. Accessed: 2014-07-06.

[67] Point Grey, "Spherical vision products." www.ptgrey.com/products/spherical.asp, 2014. Accessed: 2014-07-06.

[68] Thales Group, "Gatekeeper." http://www.thalesgroup.com/en/canada /defence/gatekeeper, 2014. Accessed: 2014-07-06.

[69] M. Avriel and D. Wilde, "Optimality proof for the symmetric Fibonacci search technique," *Fibonacci Quarterly*, vol. 4, pp. 265–269, 1966.

[70] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second ed., 2004.

[71] L. Euler, "Novi commentarii academiae scientiarum," *Petropolitane*, vol. 20, pp. 189–207, 1776.

[72] ABB, "IRB 120 industrial robot, ABB's smallest robot." `http://www.abb.co.za/productguide/product.aspx?&c=be2eef38406eaca4c125762000319182&db=seitp327`, 2014. Accessed: 2014-10-19.

[73] R. C. Gonzalez and R. E. Woods, *Digital image processing.* Addison-Wesley Publishing Company, 2002.

[74] K. A. Hawick, A. Leist, and D. P. Playne, "Parallel graph component labelling with GPUs and CUDA," *Parallel Computing*, vol. 36, no. 12, pp. 655–678, 2010.

[75] Q. Memon and S. Khan, "Camera calibration and three-dimensional world reconstruction of stereo-vision using neural networks," *International Journal of Systems Science*, vol. 32, no. 9, pp. 1155–1159, 2001.

[76] Y. Do, "Application of neural networks for stereo-camera calibration," in *International Joint Conference on Neural Networks*, vol. 4 of *IJCNN 99*, pp. 2719–2722, 1999.

[77] M. T. Ahmed, E. E. Hemayed, and A. A. Farag, "Neurocalibration: A neural network that can tell camera calibration parameters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 79, pp. 384–390, 1999.

[78] A. E. Bryson and Y. C. Ho, *Applied optimal control.* New York, USA: Blaisdell, 1969.

[79] Newport, "Kinematic bases." `http://www.newport.com/Kinematic-Bases/144476/1033/info.aspx`, 2014. Accessed: 2014-10-19.

[80] T. A. Clarke and J. G. Fryer, "The development of camera calibration methods and models," *The Photogrammetric Record*, vol. 16, no. 91, pp. 51–66, 1998.

[81] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[82] H. Bay, T. Tuytelaars, and L. V. Gool, "Surf: Speeded up robust features," in *In ECCV*, pp. 404–417, 2006.

[83] S. Leutenegger, M. Chli, and R. Y. Siegwart, "BRISK: Binary robust invariant scalable keypoints," in *International Conference on Computer Vision 2011 - ICCV2011*, 2011.

[84] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "BRIEF: Binary robust independent elementary features," in *Computer Vision–ECCV 2010*, pp. 778–792, Springer, 2010.

[85] J. Cronje, "BFROST: Binary Features from Robust Orientation Segment Tests accelerated on the GPU," in *22nd Annual Symposium of the Pattern Recognition Association of South Africa*, 2011.

[86] T. L. Saaty, *The Analytical Hierarchy Process (AHP).* New York, USA: McGraw Hill, 1980.

[87] R. Yang and M. Pollefeys, "Multi-resolution real-time stereo on commodity graphics hardware," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 1, pp. I–211–I–217 vol.1, June 2003.

[88] D. Gallup, J. M. Frahm, P. Mordohai, Q. Yang, and M. Pollefeys, "Real-time plane-sweeping stereo with multiple sweeping directions," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, June 2007.

[89] W. Xu and J. Mulligan, "Panoramic video stitching from commodity HDTV cameras," *Multimedia Syst.*, vol. 19, pp. 407–426, Oct. 2013.

[90] R. E. Miles, "On random rotations in R3," *Biometrika*, vol. 52, no. 3-4, pp. 636–639, 1965.

[91] Passmark Software, "CPU benchmarks." http://www.cpubenchmark.net, 2014. Accessed: 2014-09-16.

[92] S. Graf and T. Hanning, "Analytically solving radial distortion parameters," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2 of *CVPR2005*, pp. 1104–1109, 2005.