Vision-Guided Tracking of Complex Three-Dimensional Seams for Robotic Gas Metal Arc Welding

Maien Hamed

submitted in fulfillment of the requirements for the degree of Master of Engineering in Mechatronics in the Faculty of Engineering, the Built Environment and Information Technology at the Nelson Mandela Metropolitan University

January, 2011

Promoter: Prof T.I. van Niekerk

Co-Promoter: Prof D.G. Hattingh

Contents

List of Figures

List of Tables

Abstract

Automation of welding systems is often restricted by the requirements of spatial information of the seams to be welded. When this cannot be obtained from the design of the welded parts and maintained using accurate fixturing, the use of a seam teaching or tracking system becomes necessary. Optical seam teaching and tracking systems have many advantages compared to systems implemented with other sensor families. Direct vision promises to be a viable strategy for implementing optical seam tracking, which has been mainly done with laser vision.

The current work investigated direct vision as a strategy for optical seam teaching and tracking. A robotic vision system has been implemented, consisting of an articulated robot, a hand mounted camera and a control computer. A description of the calibration methods and the seam and feature detection and three-dimensional scene reconstruction is given.

The results showed that direct vision is a suitable strategy for seam detection and learning. A discussion of generalizing the method used as an architecture for simultanious system calibration and measurement estimation is provided.

Acknowledgements

My deepest gratitude go to my parents, and my brothers and sisters for encouraging me throughout the research for and writing of this dissertation.

This dissertation would not have been possible without the guidance, support and encouragement of my promoter, Prof Theo van Niekerk and my co-promoter, Prof Danie Hattingh. My sincerest gratitude goes to them.

It is a pleasure to thank my friends at the Automotive Components Technology Station for giving me an environment sizzling with intellectual stimulation.

Declaration

I, Maien Hamed, hereby declare that the work reported in this dissertation is my own, and that the dissertation has not previously been submitted in full or partial fulfilment of the requirements of another qualification.

Author's Signature

.....

Date:

.....

Chapter 1

Introduction

Welded joints are commonly used to assemble structures owing to their many advantages which include relatively high joint efficiency, water and air tightness, weight saving, suitability for a wide material thickness range, simple structural design and reduction in fabrication time and cost [1]. It is often possible for production companies to improve their competitiveness by automating their welding processes to improve product quality, increase production output, decrease scarp and rework and reduce labour costs [2]. Automating the welding process by means of robotic manipulators has the additional advantage of increasing process flexibility over hard automation welding machines.

Traditionally, automated welding systems made use of accurate fixtures in order to achieve the required welding accuracy to ensure adequate quality. The use of fixtures, however, increases process cycle time, increases capital cost and reduces flexibility of production systems. In addition, the use of fixtures and predetermined robot welding paths is not feasible in many situations such as in production systems with small batch sizes and in the manufacturing of large structures. As a result, seam teaching and tracking systems have been utilized to alleviate the need for accurate positioning and rigid clamping of welding

workpieces.

Seam teaching and tracking systems have been studied in research laboratories and employed in manufacturing plants using a variety of sensor families. These include contact sensors, electromagnetic sensors, ultrasonic sensors and optical sensors. Because of the many advantages of optical seam teaching and tracking systems, which include their not being limited by welded plate thickness or joint type, they have especially been widely researched and implemented. For decades, the most common strategy for performing optical seam teaching and tracking has been laser vision. Operation of laser vision seam teaching and tracking systems is based on projecting beam of laser light with known geometry (mostly a laser stripe) and imaging it with a camera whose axis is tilted with respect to that of the lasesr source (figure 1.1). Predominance of this strategy could be ascribed to its immunity to interference from welding visible electromagnetic radiation, and to its reduction of the visual information required to process. However, with the steady increase of practically available processing power and the continual advances in sensing technology, direct vision based optical seam tracking systems promise to become more attractive. They don't require the use of the relatively expensive and power intensive structured laser light source. In addition, the steadily increasing knowledge in the field streaming from continuous research and development makes additional tasks accomplishable simultaneously with seam teaching or tracking with direct vision, such as quality inspection and manipulator calibration through multi-view geometry. A possibly more important consequence of these trends is that three-dimensional scene reconstruction, which until now has been practical only in applications with pre-prepared scenes and no real-time requirements, will become feasible in robotic automated processes. What this would mean is that seam teaching and tracking could be performed on workpieces with complex three-dimensional geometries with more controlled degrees of freedom than is currently available with laser vision based systems. Systems capable of this would be useful, for

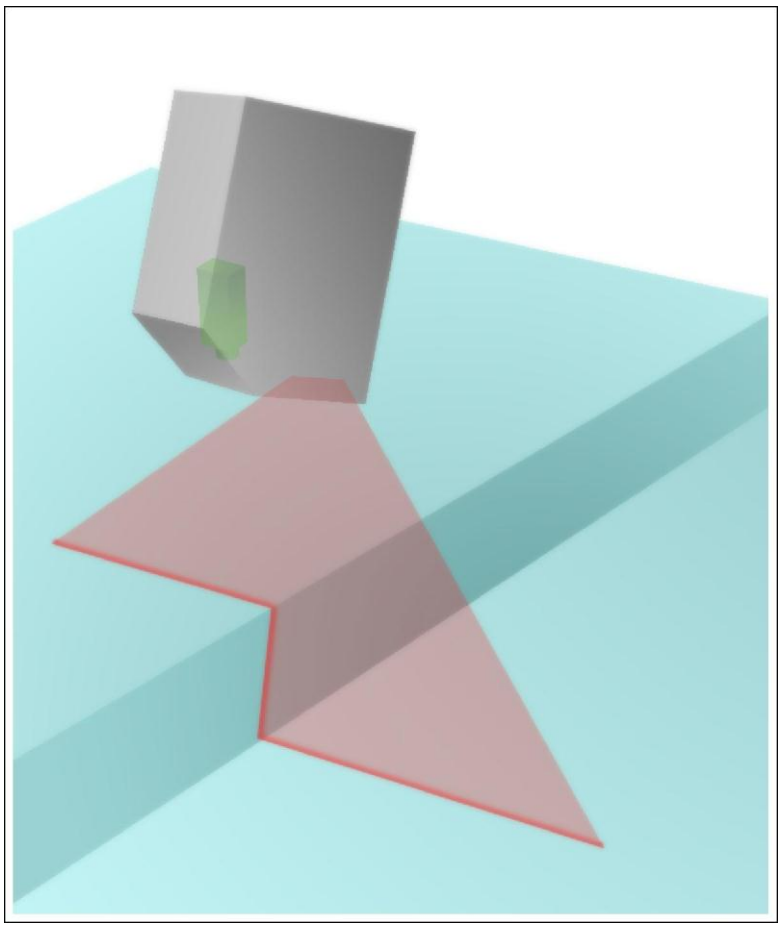


Figure 1.1: Seam Tracking by Laser Vision

instance, in performing welds on seams between intersecting curved surfaces that occur in structures such as large pipes, liquid transportation tanks, and automotive and aircraft bodies.

To investigate direct vision based seam detection and multi-view three-dimensional scene reconstruction for use in seam tracking, the main aim of this research was to develop a vision-based seam tracking system for an industrial robotic arc welding cell for welding three dimensional seams with complex geometries. The following requirements of a direct vision-based optical seam tracking system were identified:

1. Capturing images of the seam area in which the rapid changes in properties

such as orientation, reflectivity and texture that occur at the seam are enhanced

2. Detecting such rapid changes to infer the position of the seam in the images after filtering out noise that could give rise to erroneous results

3. Mapping the two dimensional information about the seam position that is obtained from the images into three-dimensional information

4. Using the three-dimensional information about the seam position to generate the paths that should be followed during the welding process.

A possible way of approaching the first requirement is by using a vision system consisting of a camera with sufficient resolution and a suitable lighting system. It is therefore necessary to select and evaluate a camera and a lighting system to be used in the investigation.

Suitable image pre-processing, edge detection and segmentation techniques must be used to fulfil the second requirement. The two dimensional information obtained by the use of these techniques may be used along with information about the camera position and orientation obtained from the robot controller to achieve the third requirement. Similarly, the three dimensional information can be used to improve the performance of the techniques used to obtain the two-dimensional information.

The fourth requirement means developing a set of rules to be used in mapping the obtained three-dimensional information of the seam into robot motion coordinates.

1.1 Objectives

Several objectives were set that contribute to reaching the goals of this research:

1. Specifying the required equipment for capturing the images to be used in the seam tracking process

2. Analyzing the geometry of welding seams and vision systems

3. Synthesising and developing algorithms for seam detection and inference of three-dimensional information through stereovision

4. Integrating, optimizing and testing a vision-based robotic weld seam tracking system to achieve the specified accuracy requirements.

After concluding this chapter with a hypothesis statement, the remaining chapters address these objectives. After a review of related literature in chapter 2, chapter 3 gives a review of computer vision and chapter 4 gives an overview of applicable mathematical theory, both of which are necessary for meeting the second and third objectives. Chapter 5 then describes the experimental setup, which addresses the first and fourth objectives and completes the third. Chapters 6 gives the experimental results, which are discussed in chapter 7.

1.2 Hypothesis

The hypothesis on which this research is based is that seams between welded surfaces exhibit rapid changes in optical characteristics that can be used for detection of seam positions in two dimensional images. In addition, this research hypothesizes that three dimensional information about the seam can be obtained from sets of two dimensional images. It is also assumed that the quality of welds can be improved by controlling welding torch following distance, orientation, speed and height over the seam.

Chapter 2

Literature Survey

Seam tracking for robotic welding is a multidisciplinary pursuit that has been researched for several decades, and has seen successful commercial implementations. However like many robotic applications, despite the apparent plateau that seam tracking systems literature has reached, seam tracking can potentially be vastly improved by applying new research findings and additional computational resources. In this chapter, a review of previous work on seam tracking (section 2.1) and welding (section 2.2) is given.

2.1 Seam Tracking

Industrial robotic welding systems have made use of various seam tracking systems in order to reduce fixturing accuracy requirements, and a large number of seam tracking research articles has been published. Several reported research works focused on non-optical seam tracking systems, such as the inductive sensor based system of Bae and Park [3], the through-the-arc sensing system of Jieyu et al. [4] and the ultrasonic sensor system of Mahajan and Figueroa [5]. How-

ever, the majority of seam tracking systems in the literature are optical. Laser vision has been a commonly used strategy of implementing optical seam tracking [6, 7, 8, 9, 10, 11]. It involves projecting a structured laser across the seam and viewing its projection with a camera whose access is angled with respect to the laser direction. Though a stripe is the most commonly used shape for the laser, use of different shapes has been reported. Xu, Tang and Yao [12], for instance, used a circular laser sensor to perform three dimensional seam tracking more effectively than can be done with a laser stripe. Bae, Lee and Ahn [13] performed seam tracking and weld pool control without structured light by capturing images at instances when a short circuit occurs in the pulsed GMAW process. This way, interference from the intensive arc light was avoided. Agapiou, Kasiouras and Serafetinides [14] presented a detailed analysis of the MIG spectrum to be used in the design and development of seam tracking sensors for GMAW processes.

2.2 Welding

Over the years, a large number of research projects on arc welding processes and their characterisation and automation have been published. The following subsections will survey various aspects of this literature.

2.2.1 Welding Characterisation

Many researchers have focused on characterisation of the gas-metal arc (GMA) and gas tungsten arc (GTA) welding processes. Their work has produced theoretical, numerical and empirical results on the characterisation of these pro-

cesses. Murugan and Parmar [15], for instance, developed a mathematical model for predicting the effect of GMAW parameters on bead geometry. Randhawa, Ghosh and Gupta [16] used a heat balance analysis to derive a qualitative understanding of the effect of pulse characteristics on bead geometry for pulsed current GMAW. Hu and Tsai [17] developed a model for heat and mass transfer and weld pool formation dynamics in GMAW. Xu, Hu and Tsai [18] derived a three dimensional model for arc plasma and metal transfer in GMAW. Goyal, Ghosh and Saini [19] studied the thermal behaviour and weld pool geometry in pulsed GMA welding. Modenesi and Reis [20] developed a numerical model for temperature distributions and melting rate in GMAW electrode wire. Ka-radeniz, Ozsarac and Yildiz [21] focused on the effect of GMAW parameters on weld penetration. They verified that weld penetration increases with current and voltage. Ghosh, Hubner and Goyal [22] used high speed video imaging to study arc characteristics and metal transfer behaviour in pulsed GMA welding. Kumar and Sundarrajan [23] empirically studied the effects of welding speed, pulse frequency and peak and base current values on ultimate tensile strength, yield strength, percent elongation and microhardness of aluminium alloys.

2.2.2 Weld Quality

Weld quality is an issue that has been addressed thoroughly in GMA research. Weld bead geometry is a criterion that is often studied as an indication of weld quality. Moon and Na [24] developed a neuro-fuzzy approach to select GMA welding current, voltage and speed for welding bead geometry improvement. Several researchers [25, 26, 27] presented mathematical models utilizing neural networks and multiple regression analysis methods to predict top bead width for improvement of robotic GMAW quality. Lee, Chang, Jang and Lee [28] studied the effect of weld geometry on fatigue strength of fillet welded joints. Several pa-

pers on weld quality directly addressed mechanical properties. Kolhe and Datta [29] studied the effects of number of passes on microstructure and mechanical properties in submerged metal arc welding (SMAW). Deng and Murakawa [30] presented a finite element analysis model for prediction of distortion and residual stresses on GMA welded thin plates. Zhang, Zhang, Cai, Gao and Wu [31] performed a numerical simulation of three-dimensional stress fields of double-side double-arc welding processes. Furthermore, several researchers have focused on techniques for quality monitoring in GMAW processes. Wang and Liao [32] presented a procedure for automatic identification of welding defects using ultrasonic imaging. Mirapeix, Garcia-Allende, Cobo, Conde and Lopez-Higuera [33] used a neural network for real-time arc defect detection using arc light spectral signals. Zhiyong, Bao, and Jingbin [34] also performed spectral analysis of arc light to detect welding quality and disturbance factors in GTAW.

2.2.3 Welding Automation

Robotic arc welding systems are common in industry and have also been the focus of much research. All the major robotic system suppliers offer arc welding robotic solutions [35, 36, 37]. Norberto Pires, Godinho and Ferreira [38] introduced a CAD interface for robotic programming of welding operations. Lauridsen, Madsen, Holm, Hafsteinsson and Boelskifte [39] presented a model based control system for a manipulator used in welding nozzles perpendicular to large diameter pipes. Lanzetta, Santochi and Tantussi [40] presented a system for online control of robotic GMAW. Their system was based on visual feedback from two charge coupled device (CCD) cameras and a laser stripe to monitor and control the weld pool and arc and inspect the bead. Smartt, Kenney and Tolle [41] presented an architecture for incorporating artificial intelligence techniques in industrial robotic welding cells. Their architecture was based on agent-based

intelligence. Bauchspiess, Absi Alfaro and Dobrzanski [42] implemented a predictive model-based control system on a sensor guided robotic welding system incorporating manipulator dynamics, sensor feedback and a path generation model. Steele, Mnich, Debrunner, Vincent and Liu [43] introduced a concept for closed loop control of robotic arc welding processes using a sensor fusion approach. Ngo, Duy, Phuong, Kim and Kim [44] developed a digital GMAW system using a decentralized control approach.

Chapter 3

Computer Vision Review

Computer vision is a wide field of knowledge that draws from the disciplines of optics, photography, computing, signal processing, cognitive science and artificial intelligence, to mention a few. The task of a computer vision system is to derive structure of a scene and other information about its environment from one or several images. This chapter contains an overview of some of the important areas of the field with focus on those that are useful in meeting the objectives of this project.

3.1 Geometrical Optics and Image Formation and Acquisition

Cornerstone to extracting useful information from any class of measurement data is understanding the underlying physical laws which govern the measured physical quantities and the measurement system used. Images, which are light intensity data measured with an imaging sensor, are no exception.

Light forms a function whose domain is position and orientation in space and whose range is intensity distribution over the visible spectrum. Considering a point in space, for instance, the intensity of light of a certain wavelength incident on that point from a specific direction at a given instance has a specific value. This value is affected by a number of variables in the environment around that point. These factors include locations and properties of light sources, media through which light travels and locations and optical properties of objects in that environments.

To form an image, light must be guided such that light intensity on a small area on the surface of an imaging sensor corresponds to light intensity on a corresponding small area of the scene. The smaller these areas, the better the focus of the resulting image. This principle underlies image formation on surfaces which light can only reach through small openings. Operation of the first cameras invented, pinhole cameras, is based on this principle. A pinhole camera is a light-proof enclosure with a pinhole on one of the walls. The term camera originates from the Latin name given to these instruments "*camera obscura*" which translates to "*dark chamber*".

Geometric optics, which is the study of light propagation in terms of rays, is an important field for all applications involving images. The laws of reflection and refraction are two of the most important results of geometric optics. They describe how light behaves when it interacts with interfaces between different media. In general, the direction of a light ray changes when it reaches the interface between different media. The ray can either be reflected such that it returns into the medium from which it originated, or refracted such that it crosses the boundary into the other medium.

The law of reflection can be used to describe reflected light rays. The reflected ray is coplanar with the incident ray and the normal vector to the surface at

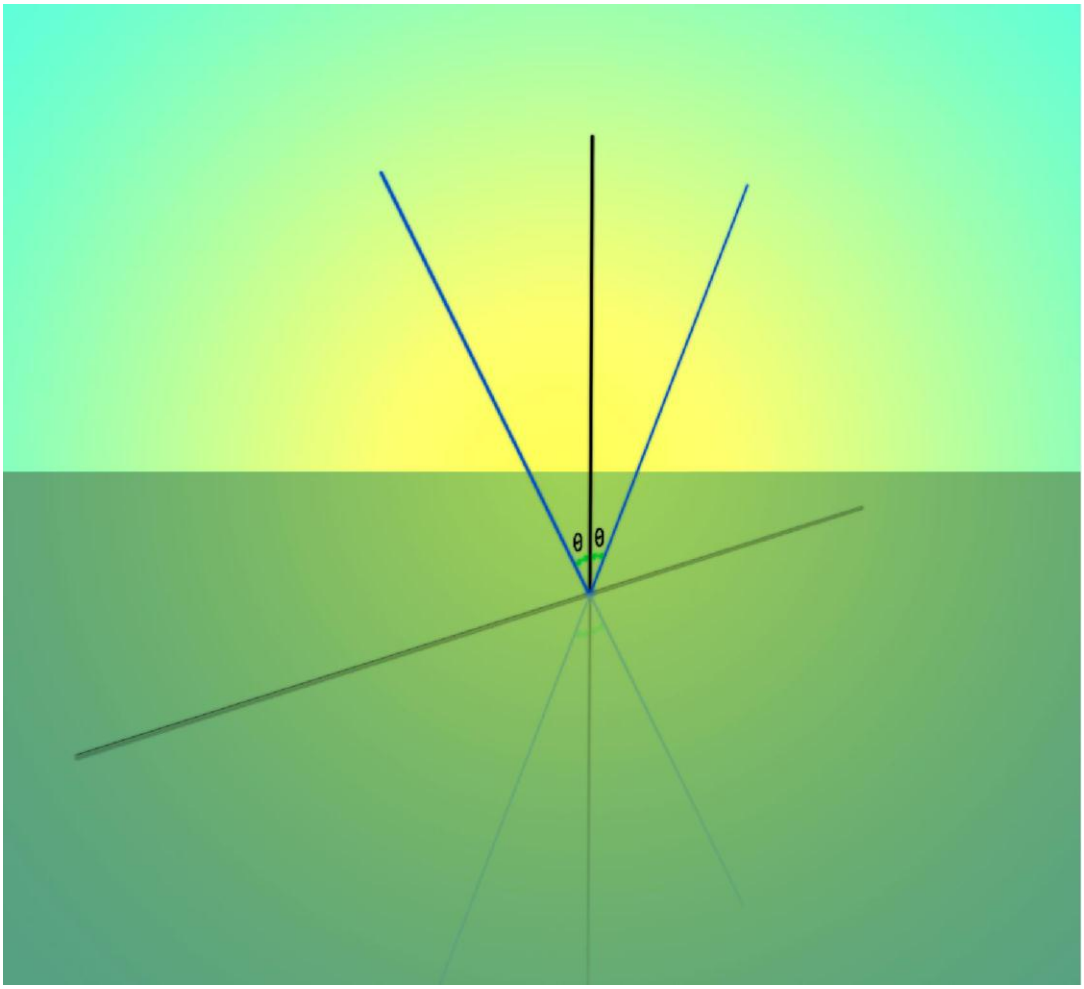


Figure 3.1: Reflection Principle

the point of reflection. Furthermore, the angle that the incident ray makes with the normal vector is equal to that between reflected ray and the normal vector (Figure 3.1).

The law of refraction describes refracted light rays. The refracted light ray is also coplanar with the incident ray and the normal to the interface surface at the point of incidence. The ratio between the sines of the angles that the incident and refracted rays make with the surface normal vector is equal to the ratio between the indices of refraction of the two media (Figure 3.2). The index of refraction of a medium is the ratio between the speed of light in that medium

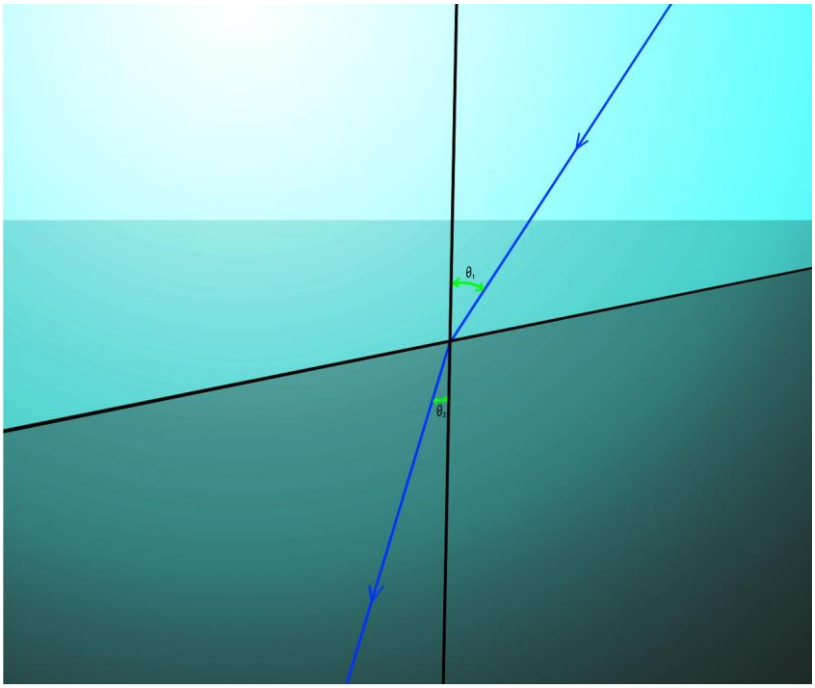


Figure 3.2: Refraction Principle

and the speed of light in free space.

The principles of geometric optics are used to design optical systems such as mirrors and lenses to guide light rays in such a way that all rays corresponding to a given point in the scene pass through one point. The set of such points to which the rays converge is the image surface. Most modern optical systems make use of lenses to form images. Lenses can be thought of as means of improvement to pinhole image systems. The main limitation of pinhole cameras is that increasing the size of the pinhole increases the amount of light on each point of the image, but makes the image blurry. On the other hand, reducing the size of the pinhole focuses the image but reduces its brightness, and therefore increases the required exposure time. Lenses act as pinholes which collect more rays of each point in the scene and focusing them on the corresponding point of the image. This allows more light from the scene onto the image while preventing the image from becoming blurred (Figure 3.3).

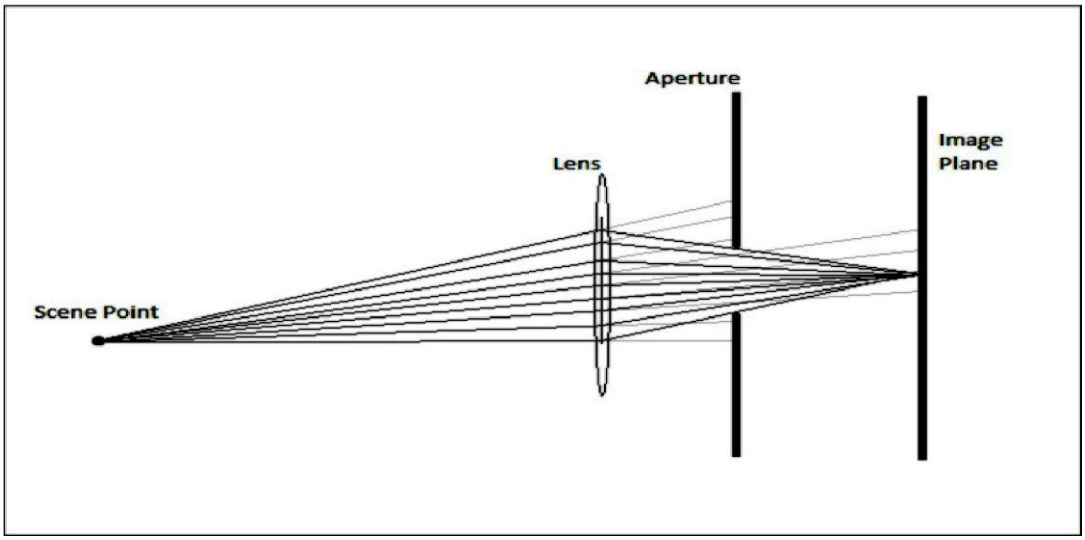


Figure 3.3: Effect of Lenses on Focus and Exposure

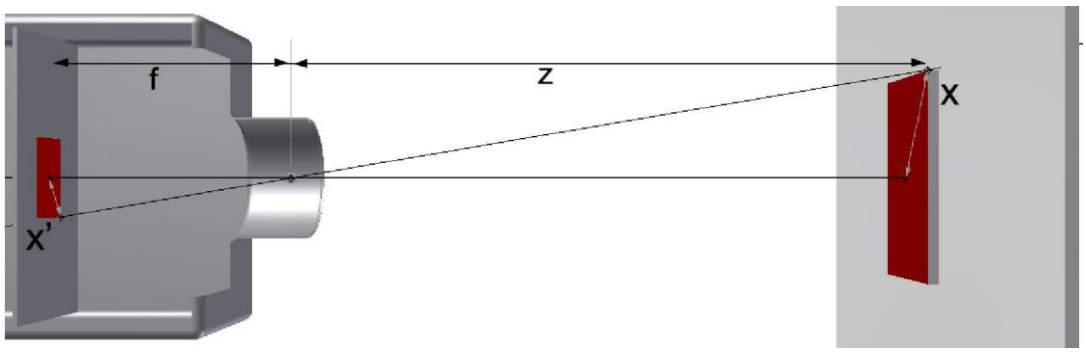


Figure 3.4: Relationship between Focal Length and Lens Magnification

The distance between the lens and the focused image is called the focal length. The focal length of a lens determines the magnification of the lens. From triangle similarity in figure 3.4, the magnification is given by:

$$M = \frac{x'}{x} = \frac{f}{z} \tag{3.1}$$

Another important parameter used in practice to describe lenses is the F–number (Also denoted F/#). It is defined as the ratio between the focal length and the

aperture diameter.

$$F/\# = \frac{f}{D} \tag{3.2}$$

The F-number is a dimensionless number that describes the light intensity that reaches the image plane. As the F-number increases, the amount of light that reaches a unit area of the image plane per unit time decreases quadratically.

In addition to light intensity at the image plane, the f-number also affects the depth of field. The depth of field is a measure of the distance range away from the lens in which objects generate sharp images. A point generates a sharp image if all light rays emanating from it fall within a small area on the image plane, which is usually specified by specifying the diameter a circle which can cover that area.

3.2 Epipolar Geometry

Epipolar geometry is a branch of geometry which describes the relationships between projections of scene points on two pinhole camera images taken at two different positions. Corresponding image points are governed by geometrical constraints which can be described by an equation involving a bilinear transformation with a matrix known as the fundamental matrix [45].

If x_L represents the image of a point in space X in any image, then X must be on the line defined by x_L and the centre of projection of that view, O_L . The projection of X on any other image is then bound to lie on the projection of

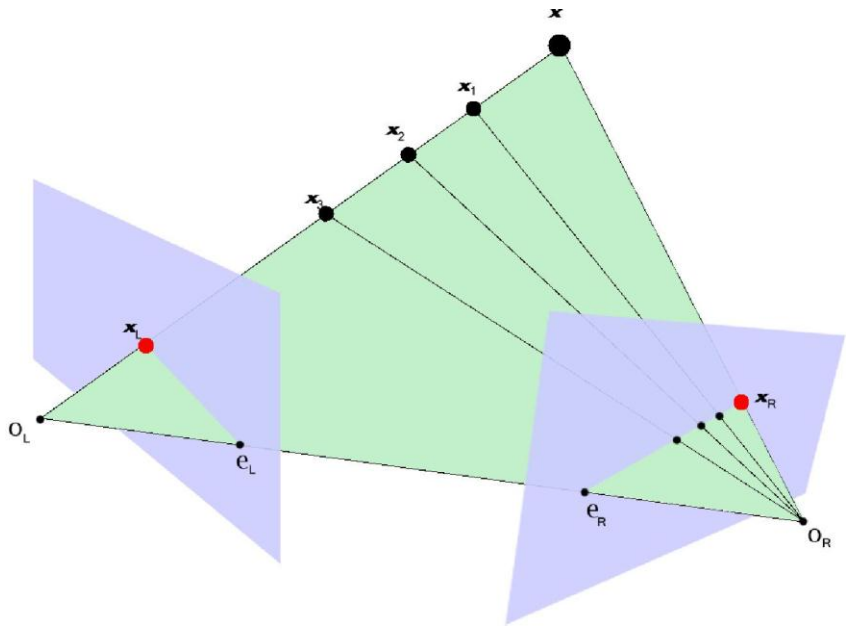


Figure 3.5: Epipolar Constraint

that line on this latter image (Figure 3.5). This observation is the premise of epipolar geometry.

The line connecting the focal points of any two views is referred to as the baseline for that stereo pair. The epipole is the intersection point of the baseline and an image plane. Any plane containing the baseline is called an epipolar plane. Epipolar planes intersect with the image planes of the two views along lines called epipolar lines. All epipolar lines in an image pass through the epipole. For any image point x_L, the corresponding point x_R on the other image lies on the epipolar line on that image which lies on the same plane as the epipolar line which contains x_L.

The fundamental matrix is an algebraic description of the constraints described

above. For a pair of images I and I', it is derived by expressing the epipolar line on I' corresponding to a point x in I as the product of two matrices $[e]_{\times}$ and H. The matrix H maps a point on I to a point on the corresponding epipolar line on I'. The matrix $[e]_{\times}$ then maps the result to the epipolar line which is the line connecting the resulting point to the epipole. The epipolar line l' corresponding to a point x on I is therefore given by:

$$l' = [e']_{\times}Hx = Fx \tag{3.3}$$

Any point x' on I' which lies on this epipolar line satisfies the equation of the line, and therefore satisfies:

$$x'^{T}l' = 0 \tag{3.4}$$

As a result, given the fundamental matrix F of a pair of images, any pair of corresponding image point pair satisfies the equation:

$$x'^{T}Fx = 0 \tag{3.5}$$

The fundamental matrix is rank deficient due to the multiplication by a cross product matrix, which is rank deficient, in its calculation. It has rank 2, and therefore the rank of its null space is 1. The basis vector of the null space of F is the homogeneous representation of the epipole in image I. The fundamental matrix has 7 degrees of freedom because it is a singular 3×3 homogeneous matrix. If F is the fundamental matrix between the image pair (I, I'), then

its transpose F^T is the fundamental matrix for the image pair in reverse order ($\dot{I}$, I).

3.3 Image Processing

Images, which are represented by scalar or vector functions of two independent variables, are the subject of image processing. The field is often divided by authors into several areas: these include image preprocessing, image segmentation, shape representation and detection, object recognition and three-dimensional vision [46, 47, 48, 49, 50]. Image preprocessing takes raw image data as input and modifies it in order to eliminate unneeded information and enhance the features in the images that are of interest to the application at hand. Preprocessing techniques are broadly divided into intensity transformations and position transformations. Image segmentation is the process of dividing an image into distinct regions which can be used for higher level algorithms such as image understanding and object recognition [46, 47, 48, 49, 50]. If information reiterated from higher level processing stages is used in segmentation, it is possible to perform complete segmentation in which each of the resulting regions corresponds to a single object in the scene. Sometimes, however, incomplete segmentation is sufficient and single objects may be divided into more than one region and single regions may have parts of more than one object [48, 49].

In some applications, the sensors used to obtain the images are distance sensors which results in images that have three-dimensional representation of the scene. In the majority of vision systems, however, pixel values correspond to light intensity rather than distance. In some applications, it is necessary to reconstruct three-dimensional information from intensity images. This is made difficult by several factors [47]. For instance, intensity images are commonly obtained by

mapping the three-dimensional world into two-dimensional space by a projective transformation in which each point corresponds to a line in space. Another factor which complicates obtaining three-dimensional information from intensity images is that the relationship between intensity and scene geometry is highly complex, with reflectivity, surface orientation, illumination and camera location all playing role in determining light intensity at a point [47]. Additional factors such as occlusion and noise also need to be addressed in three-dimensional vision applications [47]. Techniques that can be used to reconstruct three-dimensional geometry from intensity images include taking several images from different camera locations (shape from motion), taking several images with different known light source intensities and locations (shape from shading), and taking images of objects with known surface texture properties (shape from texture) [47].

3.3.1 Geometric Transformations

Geometric transformations are needed at the starting stages of image processing in a variety of situations such as when distortions in the image need to be reduced or when images of an object taken from different angles need to be compared. A geometric transformation is a vector function T that maps each point (x, y) in the image into a new location (x', y').

$$(x', y') = T(x, y) = \tilde{T_X}(x, y), \tilde{T_Y}(x, y) \qquad (3.6)$$

This function is often approximated by a m^{th}-degree polynomial of the form:

$$x' = \sum_{r=0}^{m} \sum_{k=0}^{m-r_X} a_{rk} x^r y^k \; ; \; y' = \sum_{r=0}^{X^m} \sum_{k=0}^{m-r_X} b_{rk} x^r y^k \qquad (3.7)$$

Since this transformation is linear in terms of the coefficients a_{rk} and b_{rk}, they can easily be determined from locations of several pairs of corresponding points in the image before and after transformation if they are available. Important and commonly used classes of geometrical image transformations are similarity, affine and projective transformations, which will be discussed further in section 4.7 (page 46).

Image geometric transformation functions are mostly real valued. The result of an image transformation is often required to be described in terms of a discrete raster. When that is the case, it is necessary to apply a brightness interpolation to determine the values of image points on the raster from values and locations of transformed points around them. The simplest such interpolation method is the nearest neighbor method, in which each pixel in the raster is given the value of the transformed image at the real-valued position nearest to that pixel. This method, however, is often avoided because it degrades the resultant image. The most commonly used method, which gives a smoother output, is to take a weighted average of values of image points with the weight varying with the distance from the point of interest.

3.3.2 Intensity Transformations

Transformations that depend only on pixel intensity are often applied to images as part of pre-processing or in preparation for display to human eyes. The transformed brightness of each pixel is a function of its original brightness of the form:

$$\tilde{I}(x, y) = T\,\tilde{I}(x, y) \tag{3.8}$$

3.3.3 Linear Image Transformations

Linear transformations are an important class of transformations because of the myriad of tools available for their analysis and synthesis. A linear transformation T is one under which any pair of functions $f(t)$ and $g(t)$ for any constants a and b satisfy:

$$T(a f(t) + b g(t)) = a T(f(t)) + b T(g(t)) \tag{3.9}$$

This definition of linearity holds for two dimensional images. In other words, for any images $I_1(x, y)$ and $I_2(x, y)$ and any values a and b, a linear transformation T satisfies the equation:

$$T(a I_1(x, y) + b I_2(x, y)) = a T(I_1(x, y)) + b T(I_2(x, y)) \tag{3.10}$$

Linear transformations are commonly used in signal processing and specifically in image processing. The transformed intensity of an image point is calculated as a linear function of points in the original image. An important result from linear algebra is that any linear transformation which takes as input a vector from an n-dimensional space and maps it to a vector from an m-dimensional space can be represented by multiplication by an $m \times n$ matrix [51]. This result applies to linear transformations on digitized images, since any digital image could be described as a vector with as many components as there are points in the discrete raster of the image.

3.3.4 Shift Invariant Transformations and Point Spread Functions

A translation invariant transformation, also known as a shift invariant transformation, is one which is not affected by translation of the image grid. In other words, if T is a translation invariant transformation then for any function I in the domain of T and any element h in the domain of I, it holds that:

$$I'(x) = T\,I(x) \Leftrightarrow I'(x - h) = T\,I(x - h) \tag{3.11}$$

A linear and translation invariant transformation can completely be described by the result of applying it to a point source image. In digital image processing, a point source image, often denoted $\delta(x, y)$, is one whose value is unity at the origin and zero everywhere else. The result of applying a linear translation invariant transformation to a point source image is called the point spread function of the transformation.

$$h_T(x, y) = T\,\delta(x, y) \tag{3.12}$$

Any image can be described as the sum of scaled and shifted instances of a point source image.

$$I(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j)\,\delta(x - i, y - j) \tag{3.13}$$

The result of applying a transformation T to the image is then:

$$T(I(x, y)) = T\left(\sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} I(i, j)\delta(x - i, y - j)\right) \tag{3.14}$$

Since T is a linear transformation:

$$T(I(x, y)) = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} I(i, j)T(\delta(x - i, y - j)) \tag{3.15}$$

And, recognizing the transformation of the point source image as the point spread function (Equation 3.12):

$$T(I(x, y)) = \sum_{i=-\infty}^{\infty}\sum_{j=-\infty}^{\infty} I(i, j)h_{i,j}(x - i, y - j) \tag{3.16}$$

The sum in equation 3.16 is known as the convolution of image I with the point spread function $h_{i,j}(x, y)$ of transformation T. When the image being transformed has nonzero values only within a certain region, which is usually the case, the indices of summation in the convolution formula could be changed accordingly. The point spread function of a transformation is also known as the impulse response and the kernel. Applying a linear translation invariant transformation to an image is also known in practice as filtering.

3.3.5 Image Frequency Analysis

Another useful tool in the analysis of linear transformations is the use of harmonic functions. A good starting point of developing the tools for image frequency analysis is to consider periodic functions. A function $f(x)$ is periodic with period p if it satisfies:

$$f(x + p) = f(x) \tag{3.17}$$

for every x and $(x + p)$ in its domain. The set of periodic functions with period p form an inner product space with the inner product operation defined by:

$$(f(x), g(x)) = \int_{-\frac{p}{2}}^{\frac{p}{2}} f(x)g(x)\,dx \tag{3.18}$$

Pure harmonic functions form a set of orthogonal vectors in this space, that is:

$$\int_{-\frac{p}{2}}^{\frac{p}{2}} e^{\frac{i\,2\pi k}{p}} e^{-\frac{i\,2\pi l}{p}}\,dx = 0 \qquad \text{for all } l, k \text{ with } l \neq k \tag{3.19}$$

Pure harmonics also span the entire inner product space. In other words, with the exception of the zero function, no function in the inner product space is orthogonal to the entire set of pure harmonics. As a result of these properties, a periodic function with period p can be expressed as the sum of weighted pure harmonics:

$$f(x) = \sum_{n=0}^{\infty} a_n e^{i\frac{2\pi n}{p}x}$$

$$a_n = \frac{1}{p}\int_{-\frac{p}{2}}^{\frac{p}{2}} f(x) e^{-i\frac{2\pi n}{p}x} dx \qquad (3.20)$$

This representation is known as the Fourier series representation of the function. These properties of pure harmonics also hold for two-dimensional functions such as images. The Fourier series representation of a two-dimensional function is given by:

$$f(x, y) = \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} a_{m,n} e^{2\pi i\left(\frac{mx}{M} + \frac{ny}{N}\right)}$$

$$a_{m,n} = \frac{1}{MN}\int_{-\frac{N}{2}}^{\frac{N}{2}}\int_{-\frac{M}{2}}^{\frac{M}{2}} f(x) e^{-2\pi i\left(\frac{mx}{M} + \frac{ny}{N}\right)} dx\, dy \qquad (3.21)$$

Harmonic decomposition of images is a useful tool in image processing. A direct result of linearity is that the result of applying a linear image transformation can be obtained by adding up the individual results of the linear transformation applied to the harmonic components of the image. Consequently, a linear filter can be described by its effect on harmonic images within a range of frequency values. This description is known as the frequency response of a filter.

Another useful property of the Fourier transform is that the result of multiplying the Fourier transforms of two functions is equal to the Fourier transform of the convolution of those functions.

$$F\left(f(t) * g(t)\right) = F\left(f(t)\right)F\left(g(t)\right) \qquad (3.22)$$

where $\mathcal{F}\ \tilde{f}(t)$ denotes the Fourier transform of $f(t)$ and $f(t) * g(t)$ is the convolution of $f(t)$ and $g(t)$

Since applying a linear filter to an image is equivalent to convolution of that image with the point spread function of the image (Equation 3.16), multiplying Fourier transforms of the original image and the point spread function results in the Fourier transform of the filtered image. An inverse Fourier transform can then be applied to the later to obtain the filtered image.

3.3.6 Image Smoothing

Image smoothing incorporates a number of techniques that are used to reduce noise in images. Smoothing techniques operate by assigning to each pixel in the output image a weighted average of values of pixels within some neighbourhood around that pixel in the original image. This is the same as applying a filter whose point spread function is given by the weights assigned to the pixels within the neighbourhood window. The most commonly used linear smoothing filters are the uniform filter and the Gaussian filter. Figure 3.6 shows an example of the use of image smoothing filters.

3.3.7 Edge Detection

It is often desired to find areas in an image at which the image function changes abruptly. Such edges occur at boundaries of objects or at surface brightness discontinuities within the same object caused by shadows cast thereon or changes in reflexivity. Various convolution filters could be applied to images to highlight points with high rates of change of intensity. Several filters that approximate

(a) Original Image

(b) Smoothed with Uniform Kernel

(d) Smoothed with Gaussian Kernel

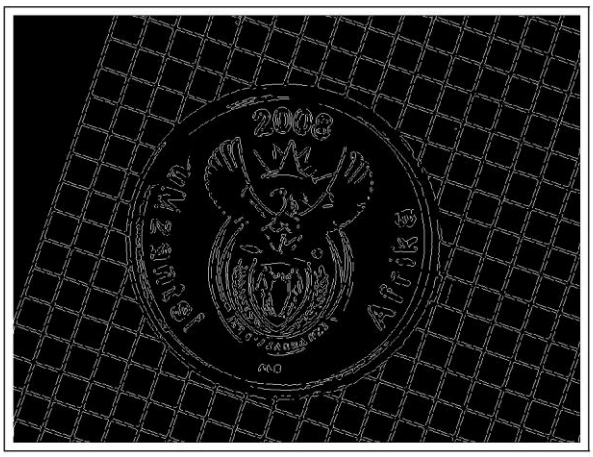
Figure 3.6: Effect of Image Smoothing

first and second derivatives of brightness were derived and have been used as edge detectors [47, 48, 49, 50]. These include the Roberts operator, the Prewitt operator and the Sobel operator which approximate the first derivative. Convolution masks which approximate the Laplacian operator which is an indirectional representation of the second derivative are also used. Canny [52] argued that derivative approximation for finding edges in images is poorly founded. As an alternative, he formulated image edge detection as a variational optimization problem. He then derived a convolution filter for edge detection that optimizes the formulated criteria. Canny's edge detector uses a derivative of a Gaussian function as the convolution filter. Figure 3.7 demonstrates the result of a Canny edge detector.

In practice, convolution with edge detection filters is followed by a refinement step which often involves thresholding the image map to classify each pixel to either belong to an edge or not [47, 48, 49, 50]. Canny's edge detection method involves using two threshold values on the premise that a pixel near an edge is more likely to be an edge pixel than one in the middle of a homogeneous brightness region [52].

A non-maximal suppression step is often part of the edge detection procedure. The aim of this step is to reduce the thickness of edges while maintaining their proximity to the actual edge in the scene. This step is performed by scanning lines perpendicular to the edge direction and removing all the edge pixels except the ones with the highest rate of change within each line [47, 48, 49, 50].

(a) Original Image



(b) Thresholded Image

Figure 3.7: Edge Detection with the Canny Edge Detector

3.3.8 Image Segmentation

Image segmentation is the process of dividing an image into distinct regions which can be used for higher level algorithms such as image understanding and object recognition. The ultimate goal of segmentation is to divide the image into segments such that there is exactly one segment for each object in the scene. This ideal case is referred to as complete segmentation. Sometimes, however, incomplete segmentation is sufficient and single objects may be divided into more than one region and single regions may contain parts of more than one

object [47, 48, 49, 501. The main image segmentation methods are:

Thresholding

Thresholding is one of the simplest methods for image segmentation. It is the oldest segmentation method and yet it is still widely used, especially in machine vision applications [471. Thresholding segmentation involves classifying pixels to belong to image objects or the background depending on whether their brightness exceeds a predetermined threshold value. Figure 3.8 demonstrates the use of thresholding for image segmentation.

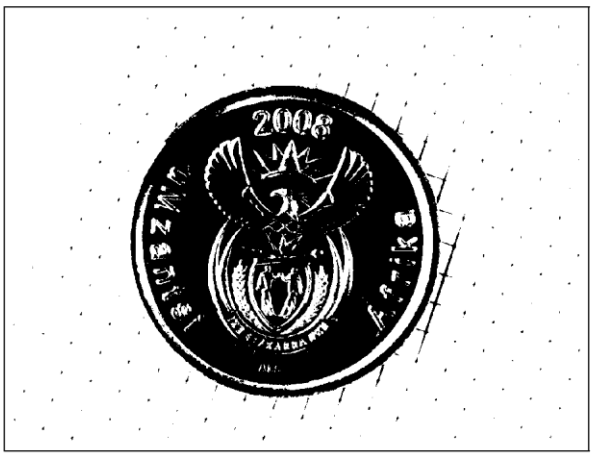
Edge-Based Segmentation

Edge-based segmentation is done by finding region boundaries by one of the edge detection methods, and then following those boundaries around each image segment.

Region-Based Segmentation

Region-based segmentation methods include region growing, region splitting or their combination. Region growing is performed by starting with several seed pixels each of which corresponding to a region in the image. Pixels surrounding the seed pixels are then added to the corresponding region until the boundaries are reached, which can be detected by a significant change in pixel properties. Region splitting is performed by starting with the image as one region and repeatedly dividing regions according to their dissimilarity.

(a) Original Image



(b) Thresholded Image

Figure 3.8: Segmentation by Thresholding

3.4 Closure

This concludes the treatment of applicable computer vision theory. An in depth, and up to date coverage of the subject is given by Szeliski [53].

Chapter 4

Mathematical Principles Review

Robotics, vision, their applications and enabling technologies have all come to the level of sophistication at which they are today mainly by describing them by mathematical models and applying old as well as relatively new mathematical techniques for their analysis and synthesis. This chapter gives an overview of some of the mathematical theory that is applicable to the subject of the current work.

4.1 Vector Spaces

A very powerful and useful concept from linear algebra is that of vector spaces. A vector space consists of a set V on which vector addition and scalar multiplication are defined such that the following conditions are satisfied for all elements u, v and w of V and all scalars s and t:

- $u + v$ is in V

- $u + v = v + u$
- $u + (v + w) = (u + v) + w$
- there is an element 0 in V such that $v + 0 = v$ for any v in V
- for any element v in V there is an element $-v$ in V such that $v + -v = 0$
- for any scalar s and any v in V, $s \cdot v$ is in V
- $s \cdot (u + v) = s \cdot u + s \cdot v$
- $(s + t) \cdot v = s \cdot v + t \cdot v$ and $s \cdot (t \cdot v) = (s\ t) \cdot v$
- there is a scalar 1 such that $1 \cdot v = v$ for any v in V

4.1.1 Inner Product Spaces

An important class of vector spaces is inner products spaces. The vector space described above is also an inner product space if, in addition to the conditions above, there is an operation which maps any two vectors u and v into $< u, v >$ such that for any elements u and v of V and for any scalars s and t:

- $<u, v> = <v, u>$
- $<u, sv + tw> = s <u, v> + t <u, w>$

- $< u, u > > 0$ and $< u, u > = 0$ if and only if $u = 0$

If V is an inner product space, the geometrical concepts of angle and length are defined using the inner product operation. The length, or norm, of a vector v is given by:

$$11v11 = \sqrt{<v, v>} \quad (4.1)$$

and the angle between two vectors u and v is given by:

$$\cos\theta = \frac{<u, v>}{11u111M11} \quad (4.2)$$

This also gives a definition for orthogonality between vectors in inner product spaces. Two nonzero vectors u and v are orthogonal if and only if $< u, v > = 0$.

Two more useful definitions are those of an orthogonal set of vectors and an orthonormal set of vectors. An orthogonal set of vectors is one in which all pairs of distinct vectors are orthogonal. An orthonormal set of vectors is an orthogonal set each of whose elements have a norm of 1.

4.1.2 Basis Vectors and Dimension of a Vector Space

A set of vectors $\{x_1, x_2, \cdots, x_n\}$ is said to span a vector space V if any vector v in V can be represented as:

$$v = a_1x_1 + a_2x_2 + \cdots + a_nx_n \quad (4.3)$$

for some scalars $(a_1, a_2, \cdots, a_n)$. If, in addition to spanning V, the set is linearly independent, then the set $\{x_1, x_2, \cdots, x_n\}$ is called a basis for the vector space V. The number of vectors in the set, n, is the minimum number of vectors

required to span V, and is the maximum number of vectors in a linearly independent subset of V. This number is called the dimension of V, and V is said to be an n-dimensional vector space.

For some vector spaces, such as the set of all functions defined in $(-\infty, \infty)$, no finite subset spans the entire space. Such vector spaces are called infinite-dimensional.

4.1.3 Cross Product

A useful operation on R^3 is that of the cross product. It takes as input two 3-dimensional vectors, v_1 and v_2, and gives as output a 3-dimensional vector $v_1 \times v_2$ which is perpendicular to the plane containing v_1 and v_2 and whose length is:

$$\|v_1 \times v_2\| = \|v_1\|\|v_2\| \sin\theta \tag{4.4}$$

where θ is the angle measured from v_1 to v_2 (Figure 4.1). The cross product of two vectors $v_1 = [v_{1x}, v_{1y}, v_{1z}]^T$ and $v_2 = [v_{2x}, v_{2y}, v_{2z}]^T$ is given by:

$$v_1 \times v_2 = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ v_{1x} & v_{1y} & v_{1z} \\ v_{2x} & v_{2y} & v_{2z} \end{vmatrix} \tag{4.5}$$

where $\hat{i}$, $\hat{j}$ and $\hat{k}$ are the unit vectors in the directions of the three Cartesian axes in three-space, and $|M|$ represents the determanent of M. Another way of obtaining the cross product is by representing it as multiplication by a skew symmetric matrix, which is called a cross product operator. The previous cross product, for instance, can be written as:

$$v_1 \times v_2 = \begin{bmatrix} 0 & -v_{1z} & v_{1y} \\ v_{1z} & 0 & -v_{1x} \\ -v_{1y} & v_{1x} & 0 \end{bmatrix} \begin{bmatrix} v_{2x} \\ v_{2y} \\ v_{2z} \end{bmatrix} \tag{4.6}$$

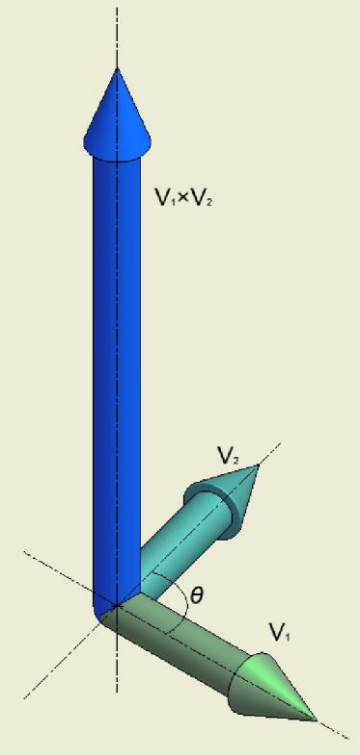


Figure 4.1: Cross Product

The cross product operator corresponding to a vector v is denoted $v\times$.

4.2 Linear Transformations

Linear transformations play an important role in computer vision and robotics. They are a vital tool for geometry computations, dynamics and control, and image processing (Subsection 3.3.3 on page 20). A general linear transformation T is one which satisfies:

$$T(u + v) = T(u) + T(v) \text{ and } T(cv) = cT(v) \tag{4.7}$$

for any u and v in its domain and any scalar c.

A linear transformation which maps vectors from an n-dimensional space to

ones in an *m*-dimensional vector space can be represented by a *m* x *n* matrix multiplication of the form:

$$\begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_m \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \cdots \\ x_n \end{bmatrix} \tag{4.8}$$

or, written in symbols:

$$y = Ax \tag{4.9}$$

In this transformation, the output y is a weighted sum of the columns of A, the weights being simply the components of the input x. The range of the linear transformation is the set of the outputs produced by the transformation for all possible inputs. Since the outputs are a linear combination of the columns of the transformation matrix, the range of the transformation is simply the subspace spanned by the vectors whose components are given by the columns of the transformation matrix. An immediate result is that the dimension of the output space is at most as large as the number of columns. If the column vectors are not linearly independent, then the dimension of the output space is as large as the largest subset of linearly independent columns. This number is unique, and it is also the size of the largest subset of linearly independent rows. This number an important property of the matrix, as well as the linear transformation, and is called the rank of the matrix, and also the rank of the transformation.

4.3 Quadratic Forms

A quadratic form is a polynomial all of whose terms with nonzero coefficients are of the second degree. A quadratic form of a set of variables $x_1, x_2, \ldots, x_n$ is

a polynomial of the form:

$$q = a_{11}x_1^2 + a_{12}x_1x_2 + \ldots + a_{1n}x_1x_n + a_{22}x_2^2 + a_{23}x_2x_3 + \ldots + a_{nn}x_n^2 \quad (4.10)$$

This quadratic form can be written as the matrix multiplication:

$$q = [x_1\ x_2\ \ldots\ x_n] \begin{bmatrix} a_{11} & \frac{1}{2}a_{12} & \ldots & \frac{1}{2}a_{1n} \\ \frac{1}{2}a_{12} & a_{22} & \ldots & \frac{1}{2}a_{2n} \\ \ldots & \ldots & \ldots & \ldots \\ \frac{1}{2}a_{1n} & \frac{1}{2}a_{2n} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} \quad (4.11)$$

By defining x as the column vector of the variables $x_1, x_2, \ldots, x_n$ and A as the matrix of coefficients, the quadratic form (4.11) can be written as:

$$q = x^T \mathrm{A} x \quad (4.12)$$

The matrix A does not have to be symmetric, but any quadratic form can be represented by a symmetric matrix [51].

4.3.1 Positive Definiteness

A useful class of quadratic forms is positive definite quadratic forms. A positive definite quadratic form A is one which satisfies:

$$x^T \mathrm{A} x > 0 \text{ for all non-zero vectors } x \quad (4.13)$$

4.4 Matrix Decomposition

In this section, several techniques for taking a matrix which describes a linear transformation or a quadratic form and expressing it as a product of matrices

with useful properties [51].

4.4.1 QR Decomposition

An $m \times n$ matrix A with columns $\{a_1, a_2, \bullet\bullet\bullet, a_n\}$ can be expressed as the product of an $m \times n$ matrix Q whose columns $\{q_1, q_2, \bullet\bullet\bullet, q_n\}$ form an orthonormal set and an invertible upper triangular matrix R (Equation 4.14).

$$A = QR \tag{4.14}$$

4.4.2 Eigenvalues and Eigenvectors

An Eigenvector of a square matrix A is a nonzero vector v which satisfies the equation:

$$Av = \lambda v \tag{4.15}$$

for some scalar λ, which is said to be an Eigenvalue of A. In words, the transformation described by A does not change the direction of v, but simply scales it by λ.

Given a set of m Eigenvectors of a matrix A, the resulting expressions of the form in equation 4.15 can be combined in the matrix form:

$$AP = P\Lambda \tag{4.16}$$

where

$$\sim P = \begin{vmatrix} & & & \\ v1 & v2 & \bullet\cdot\cdot & \sim vm \\ & & & \end{vmatrix} \tag{4.17}$$

and

$$\Lambda = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix}$$

If $\{v_1, v_2, \ldots, v_n\}$ are n linearly independent Eigenvectors of a n x n matrix A, using them to form a matrix P of the form in equation 4.17 above results in an invertible matrix. Post multiplying both sides of equation 4.16 by P^{-1} results in the Eigenvalue decomposition of A:

$$A = P\Lambda P^{-1} \tag{4.18}$$

Furthermore, if A is a n x n symmetric matrix, then it has real eigenvalues and a set of n orthogonal Eigenvectors. In other words, an Eigenvector matrix P can be formed such that $P^{-1} = P^T$ and the Eigenvalue decomposition of A becomes:

$$A = P\Lambda P^T \tag{4.19}$$

4.4.3 Singular Value Decomposition

An m x n matrix A can always be factorized to the form:

$$A = USV^T \tag{4.20}$$

with U being an m x m orthonormal matrix, V an n x n orthonormal matrix and S a unique m x n diagonal matrix whose diagonal entries are ordered in descending order. The entries of S are called the singular values of A.

Singular value decomposition can be thought of as a process of decomposing the transformation represented by A into three transformations. The first transformation, which is represented by V^T is a rotation of the input vector. The second transfomration represented by S is a scaling of the components of the input vector along the directions of the columns of V weighted by the diagonal entries of S. The third and last transformation, which is represented by U is a rotation of the resulting vector in the output space.

A property of a matrix A which can be easily extracted from its singular value decomposition (*SVD*) is its spectral norm. The spectral norm of a matrix is the maximum ratio between the output and input vectors of the transformation which it represents. That is:

$$\|A\| = \max_{x \neq 0} \frac{\|Ax\|}{\|x\|} \tag{4.21}$$

Since neither U nore V change the length of a vector because they both are orthonormal, the only change in length happens in the middle transformation of scaling the vector by the entries of S. As a result, the spectral norm of the matrix is simply its largest singular value. Moreover, the first column of V gives the direction of vectors which maximize the ratio in equation 4.21.

Another useful description of the matrix which can be extracted from its SVD is the condition number. It is the ratio of the largest singular value and the lowest non-zero singular value of the matrix. The condition number is a measure of stability.

The singular value decomposition of a matrix A can also be used to find its

pseudo-inverse $A^{\dagger}$ using the formula:

$$A_{\dagger} = V S^{\dagger} U^{T} \tag{4.22}$$

where the pseudo-inverse of the diagonal matrix S, $S^{\dagger}$, is obtained by taking the reciprocal of each non-zero entry of S and transposing it. The linear system, $Ax = b$, has a least squares solution if it is overdetermined, or a least norm solution if it is underdetermined, given by:

$$\tilde{x} = A^{\dagger} b \tag{4.23}$$

4.5 Least Squares Solutions

Linear systems of the form $Ax = b$ with A being an $m \times n$ matrix and $m > n$ arise frequently. In general, there is no exact solutions x which satisfy the equation for known A and b. An approximate solution is saught which minimizes the risidual, which is defined as the sum of squares of error values:

$$r = \|Ax - b\| \tag{4.24}$$

Such a solution, often denoted $\tilde{x}$, is the least squares solution. It can be thought of as the components of the projection of the m-dimensional vector b onto the n-dimensional column space of A in terms of the columns of A. In other words, the error ($A\tilde{x} - b$) must be orthogonal to all the columns of A. Therefore:

$$A_T (A\tilde{x} - b) = 0 \tag{4.25}$$

which can be re-arranged to yield the normal equation:

$$\tilde{x} = (A^T A)^{-1} A^T b \tag{4.26}$$

when the inverse of $A^T A$ exists, which is true for a full rank $m \times n$ matrix with $m > n$.

The least squares solution can also be found using the singular value decomposition as discussed in subsection 4.4.3.

4.6 Analytic Geometry

Geometric entities can be treated as algebraic objects by describing them in terms of coordinate systems. This approach of studying geometry is known as analytic geometry. By defining a cartesian coordinate system, for instance, any point can be described by the vector that connects the origin of the coordinate system to that point. Motions can then be described as vector transformations, and curves and surfaces can be described as sets of vectors. The following subsections elaborate on this concept.

4.6.1 Curves and Surfaces in Space

Curves and surfaces are collections of points in space. They can be described by the collection of vectors to those points in a coordinate system defined in their space.

For instance, a line can be represented as the set of points located at a specific direction from some point, x_o, on the line. The direction is determined by the vector connecting x_o to any other point x_i on the line. Any point on the line, for some $t \in \mathbb{R}$, satisfies:

$$x = x_o + t(x_i - x_o) \tag{4.27}$$

which can be rearranged as:

$$x = (1 - t)x_o + tx_i \tag{4.28}$$

The line, then, is the set $\{x = (1 - t)x_0 + tx_1 : t \in \mathbb{R}\}$. Curves or surfaces are often expressed by an equation exclusively satisfied by all the points which they contain. Often, the set notation is dropped and the geometrical object is described by the equation alone.

A plane can be described using its normal vector and any point, x_0, that it contains. The normal vector, by definition, is orthogonal to any vector connecting x_0 to any point on the plane. As a result, the inner product of the corresponding vectors is 0:

$$(x - x_0, n) = 0 \tag{4.29}$$

If x_0 is $[o_x, o_y, o_z]$ and n is $[a, b, c]$, then the equation of the plane becomes:

$$a(x - o_x) + b(y - o_y) + c(z - o_z) = 0 \tag{4.30}$$

Another way of describing curves and surfaces is giving a parametric representation of their coordinates. For instance, points on the line described by equation 4.28 have the form:

$$x = \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} (1 - t)x_0 + t x_1 \\ (1 - t)y_0 + t y_1 \\ (1 - t)z_0 + t z_1 \end{bmatrix} \quad t \in \mathbb{R} \tag{4.31}$$

In general, a curve can be described as a point, x, whose components are given by functions of a single parameter, t:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x(t) \\ f_y(t) \\ f_z(t) \end{bmatrix} \quad t \in \mathbb{R} \tag{4.32}$$

A surface, similarly, can be represented by points whose components are functions of two independent parameters, s and t:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} f_x(s, t) \\ f_y(s, t) \\ f_z(s, t) \end{bmatrix} \quad s, t \in \mathbb{R} \tag{4.33}$$

Directional Differentiation

Parametric representation makes the use of the tools of differential and integral calculus possible for the analysis of curves and surfaces. The slope of a curve or a surface at any point and direction can be found, if it exists, using partial differentiation. Differentiating the parametric equation of a curve or a surface (Equations 4.33 and 4.32) with respect to a certain parameter produces the rates of change of each of the coordinates with respect to that parameter. This can be used for finding the slope of a curve or a surface in any direction. Carrying out the differentiation in terms of one of the coordinates results in the rate of change in the direction of the corresponding axis.

Another problem that can readily be solved by partial differentiation is finding the plane tangent to a surface represented by equation (4.33) at any point $\mathbf{x}_0$. Differentiating the parametric equation in terms of each of the two parameters and evaluating the derivatives at $\mathbf{x}_0$ produces two independent direction vectors on the surface. Any point on the surface can be represented by the sum of the vector to $\mathbf{x}_0$ and multiples of these direction vectors. Moreover, the normal to the surface can be obtained from the cross product of the two direction vectors.

If a curve in space represented in the form of equation 4.32 is to be traversed at a constant speed, the parameter needs to be increased at a rate which is inversely proportional to the rate of change of the length along the curve with respect to the parameter t. The later is given by:

$$\frac{ds}{dt} = \sqrt{\left(\frac{df_x}{dt}\right)^2 + \left(\frac{df_y}{dt}\right)^2 + \left(\frac{df_z}{dt}\right)^2} \tag{4.34}$$

Taylor Expansion and Linearization

The Taylor expansion of an infinitely differentiable function of a single real variable is given by:

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}}{n!}(x - x_0)^n \tag{4.35}$$

where $f^{(n)}$ is the nth derivative of f. This applies directly to parametric curves in space. A curve in space given by equation (4.32), for instance, has the Taylor expansion:

$$x(t) = \begin{pmatrix} \sum_{n=0}^{\infty} \frac{f_x^{(n)}(t_0)}{n!}(t-t_0)^n \\ \sum_{n=0}^{\infty} \frac{f_y^{(n)}(t_0)}{n!}(t-t_0)^n \\ \sum_{n=0}^{\infty} \frac{f_z^{(n)}(t_0)}{n!}(t-t_0)^n \end{pmatrix} = \sum_{n=0}^{\infty} \begin{pmatrix} \frac{f_x^{(n)}(t_0)}{n!} \\ \frac{f_y^{(n)}(t_0)}{n!} \\ \frac{f_z^{(n)}(t_0)}{n!} \end{pmatrix} (t-t_0)^n \tag{4.36}$$

Linearization of a function means finding the linear function which most closely approximates it. That can be done by taking the linear terms of the Taylor expansion. Linearization of the same parametric curve above is given by:

$$x_L(t) = \begin{pmatrix} f_x(t_0) \\ f_y(t_0) \\ f_z(t_0) \end{pmatrix} + \begin{pmatrix} f_x'(t_0) \\ f_y'(t_0) \\ f_z'(t_0) \end{pmatrix} (t-t_0) \tag{4.37}$$

The concept of Taylor expansion can be extended to surfaces in space and general functions of multiple real parameters by taking partial derivatives in terms

of all the possible parameters. Linearization of such multivariable functions around a point $\boldsymbol{T}_0 = [s_0, t_0, \cdots]^T$ in the parameter space is given, similarly, by:

$$\boldsymbol{X}_L(s, t, \ldots) = \begin{bmatrix} f_x(\boldsymbol{T}_0) \\ f_y(\boldsymbol{T}_0) \\ f_z(\boldsymbol{T}_0) \end{bmatrix} + \begin{bmatrix} \frac{df_x}{ds}\Big|_{r_0} & \frac{df_x}{dt}\Big|_{r_0} & \cdots \\ \frac{df_y}{ds}\Big|_{r_0} & \frac{df_y}{dt}\Big|_{r_0} & \cdots \\ \frac{df_z}{ds}\Big|_{r_0} & \frac{df_z}{dt}\Big|_{r_0} & \cdots \end{bmatrix} \begin{bmatrix} (s - s_0) \\ (t - t_0) \\ \ldots \end{bmatrix} \tag{4.38}$$

4.7 Homogeneous Coordinates and Projective Geometry

The two-dimensional and three-dimensional Euclidian spaces can be extended to projective spaces by introducing an additional coordinate to the representation of vectors. A vector in a Euclidean plane with the coordinates $[x, y]^T$ is represented as $[x, y, 1]^T$ in the projective space. The representation $[x, y, 1]^T$ is equivalent to $[ax, ay, a]^T$ for any nonzero a. Similarly, a three-dimensional vector with coordinates $[x, y, z]^T$ in a Euclidean space has a homogeneous representation $[x, y, z, 1]^T$ which is equivalent to $[ax, ay, az, a]^T$ for any nonzero a. Representing vectors in homogeneous coordinates enables non-singular linear transformations to represent a more general family of transformations than is possible if they act on standard Euclidean vectors [45, 2]. This makes homogeneous vectors and projective transformations useful in robotics and vision since they represent important processes that cannot be represented by Euclidean transformations such as translation of coordinates and perspective projection. To clarify, a linear transformation in a three-dimensional projective space can

be represented by a 4 × 4 matrix with the blocked representation [2, 45]:

$$T_{projective} = \begin{bmatrix} R_{3x3} & t_{3x1} \\ 77_{T}{}_{3x1} & \sigma \end{bmatrix} \tag{4.39}$$

The matrix R_{3x3} is a rotation matrix. In robotics, it is always an orthogonal matrix which represents coordinate frame rotation. In vision and computer graphics, R_{3x3} represents both rotation and scaling of axes and is therefore not necessarily orthogonal. The vector t_{3x1} represents translation of the origin of a coordinate frame. The vector $_{77}3_{x1}$ represents perspective scaling. And the scalar σ is a scaling factor for the transformation.

Another useful consequence of representing vectors with their homogeneous coordinates is that it gives rise to a more general representation of intersection of lines and planes. In plane geometry, a line can be represented by an equation:

$$ax + by + c = 0 \tag{4.40}$$

A line given by equation 4.40 can be represented by the vector $[a, b, c]^T$. A point $[x_0, y_0]$ is on the line if and only if it satisfies equation 4.40 and we can say:

$$ax_0 + by_0 + c = 0 \tag{4.41}$$

By multiplying both sides of the equation by a non-zero scalar σ, it can be seen

that:

$$a(\sigma x_0) + b(\sigma y_0) + c(\sigma) = 0 \tag{4.42}$$

In words, a point with homogeneous coordinates $[x_0, y_0, 1]^T$ is on a line represented by a vector $[a, b, c]$ if and only if the two vector representations are orthogonal. As a result, two lines intersect at a point if and only if the vector with the homogeneous coordinates of that point is orthogonal to both the vectors given by the coordinate representations of the two lines. Similarly, a line connects two points if and only if the vector given by its coordinate representation is orthogonal to both the vectors given by the sets of homogeneous coordinates of the two points. Because of this fact and since the cross product of two vectors is orthogonal to both, the cross product becomes a useful tool in finding intersection points of lines and lines connecting points on a plane.

A similar reasoning can be followed by using the vector $[a, b, c, d]$ to represent a plane in a three-dimensional projective space given by the equation:

$$ax + by + cz + d = 0 \tag{4.43}$$

A point with homogeneous coordinates $[x_0, y_0, z_0, 1]^T$ is on this plane if and only if the inner product of the two vectors representing the point and the plane evaluates to zero. This observation can used to show that a point where three planes intersect or the plane defined by three points is defined by any vector in the null space of the matrix whose rows are formed by the homogeneous representations of the planes or the points respectively.

4.8 Closure

This concludes the review of the applicable mathematical principles. The principles discussed are utilized at various subsystems in the experimental setup, which is discussed in the following chapter.

Chapter 5

Experimental System

The main component of the experimental system was a charge coupled device (CCD) camera mounted on a bracket attached to the flange of a 6-axis industrial robotic manipulator. Images taken by the camera were transferred to a control computer through an IEEE 1394a (FireWire) link. The task of the control computer was to process the images it received through the software developed in this project. The control computer was connected to the controller of the industrial robot through an Ethernet link which conveyed position feedback and motion commands between the robot and the computer. The experimental setup also comprised a section of a cylindrical stainless steel tank onto which circular flanges were welded. Figure 5.1 shows a schematic representation of the system organization, and figure 5.2 shows a photograph of the system.

Figure 5.3 gives a schematic representation of the software architecture. As shown in the figure, images captured by the camera are first pre-processed by smoothing and enhancement as described in chapter 3. The resulting images are then processed for seam and feature detection (section 5.4). The detected and matched features are then used for scene three-dimensional geometry reconstruction through bundle adjustment (section 5.5), which gives the three-dimensional

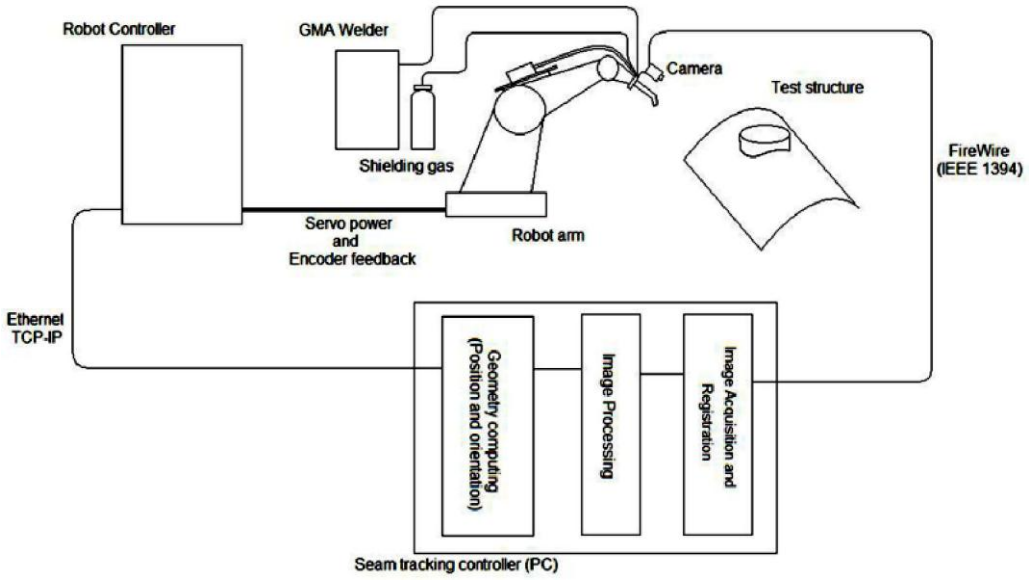


Figure 5.1: System Architecture

seam geometry required for tracking. Before seam and feature detection and bundle adjustment are described, however, some aspects of the robot used are addressed in section 5.1, and a detailed description of the process followed for calibrating the camera is given in section 5.3

5.1 Robot Arm and Controller

The robot manipulator used in this study was a 6-axis KUKA KR30/2 manipulator. It is rated for a 30 kg payload and a ±0.15 mm repeatability. To integrate the seam tracking system with the robot, three issues needed to be addressed: mounting the camera on the robot (mechanical interfacing: subsection 5.1.1), processing position feedback into usable coordinate transformations (geometrical interfacing: subsection 5.1.2) and retrieving this position feedback and

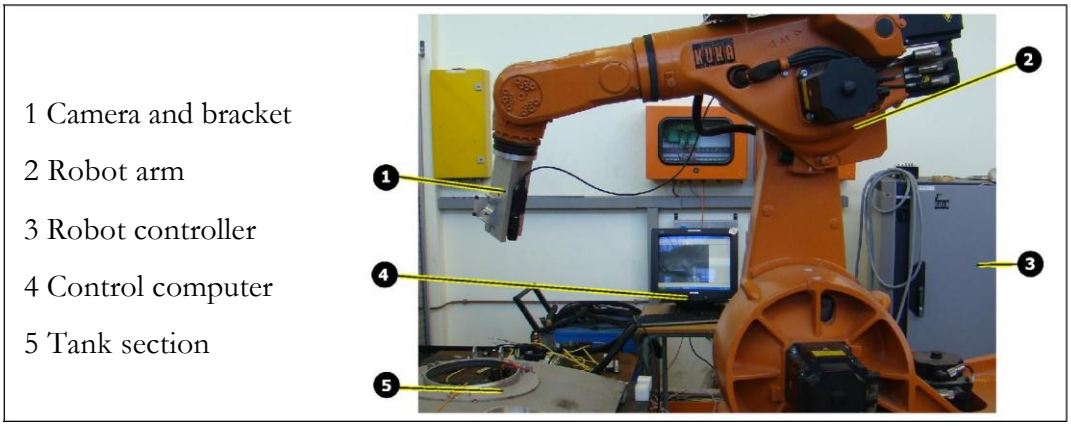


Figure 5.2: Experimental System

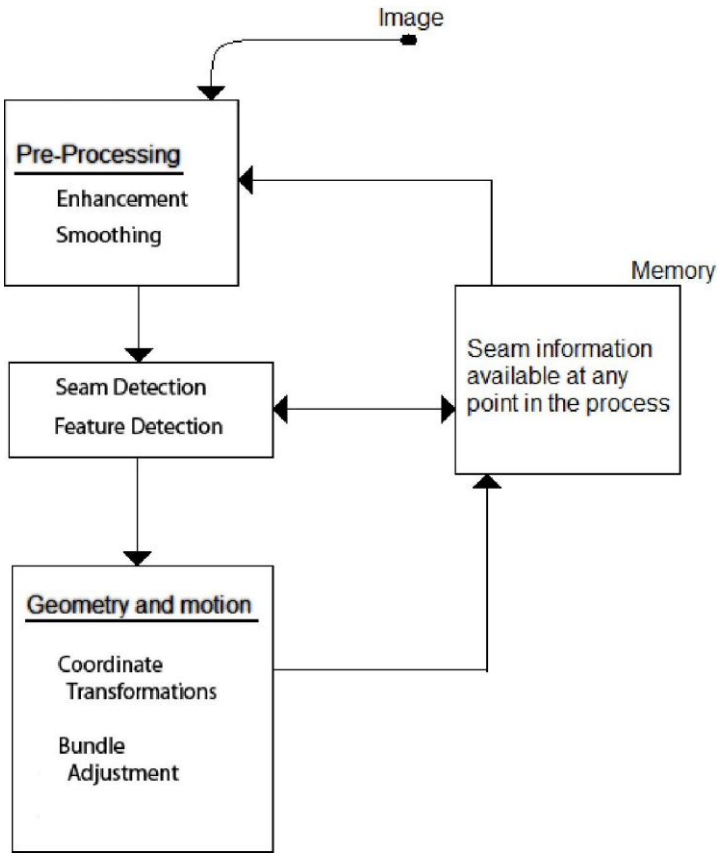


Figure 5.3: System Software Architecture

communicating motion commands with the robot (network interfacing: subsection 5.1.3).

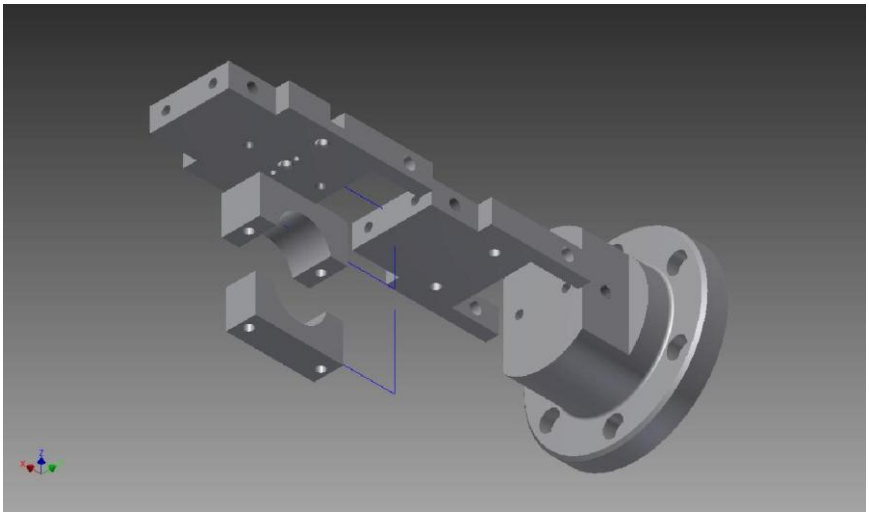
5.1.1 End Effector (Mechanical Interface)

During the seam tracking tests, the robot was required to move the camera, the welding torch and possibly a structured light source. A bracket was designed to be mounted on the flange of the robot to act as a mechanical interface for these objects.

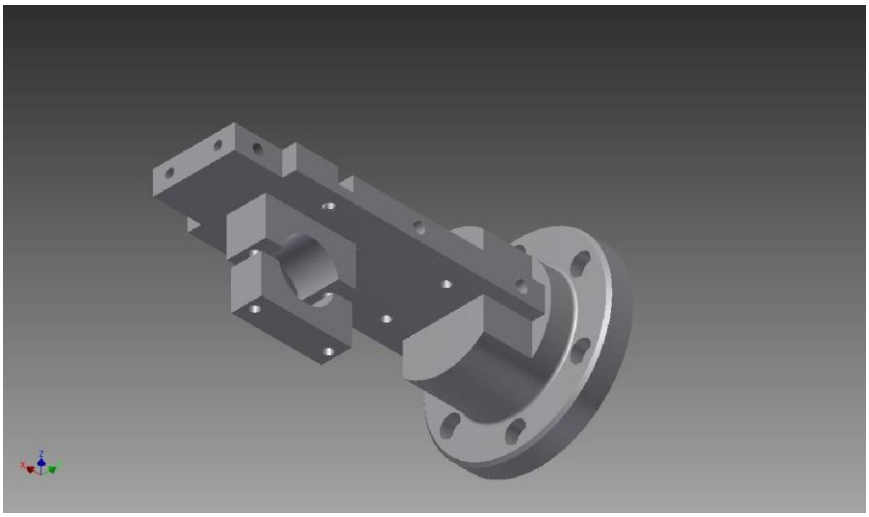
Two concepts were considered in the initial stage of the design. The first concept was a modular design which allowed addition of mounting interfaces by adding links with pre-defined interconnection geometry. This concept is shown in figure 5.4. The second concept design featured two plates bolted on a flange as shown in figure 5.5. The second concept was chosen for this project because it was easier to manufacture and it promised higher rigidity, which was considered critical for the purposes of this investigation. Figure 5.6 shows a photograph of the manufactured and assembled bracket.

5.1.2 Coordinate Transformation from Position Feedback

The KRC2 robot controller provides feedback about the position of the end effector by means of providing 6 values corresponding to the position and orientation of the three axes of the tool coordinate frame. These parameters correspond to the three components of the translation vector from the origin of a programmable base frame and the roll, pitch and yaw angles with respect to that frame. In order to transform points between the tool coordinate frame

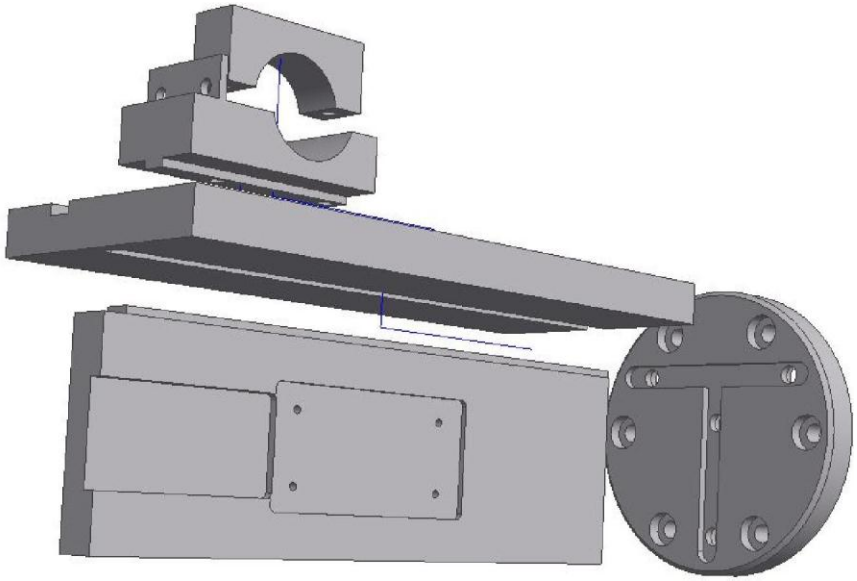


(a) Exploded View

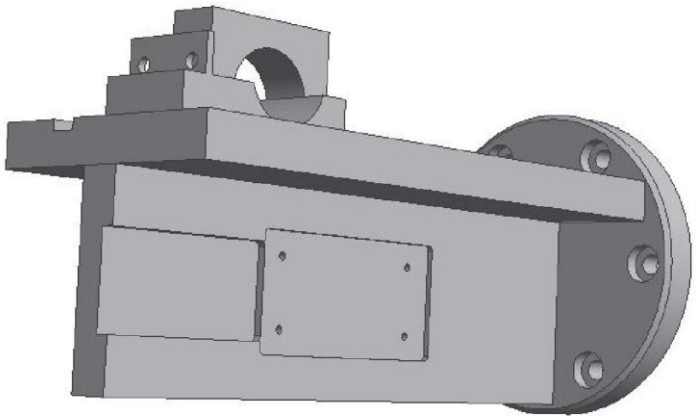


(b) Assembled View

Figure 5.4: Modular Mechanical Interface Concept



(a) Exploded View



(b) Assembled View

Figure 5.5: Plate-Based Interface Concept

Figure 5.6: Manufactured Bracket with the Camera Mounted Thereon

and the camera and world coordinate frames, these parameters must be used to obtain the appropriate transformation matrices. A commonly used notation in robotics texts is to abbreviate the symbols for the trigonometric functions of an angle *0* as:

$$\sin 0 = s0$$

$$\cos 0 = c0$$

Using this notation, the transformation from a base coordinate frame to the tool coordinate frame is given by [2]:

$$ {}^{t}T_{b} = \begin{bmatrix} cAcB & sAcB & -sB & 0 \\ cAsBsC - sAcC & sAsBsC + cAcC & cBsC & 0 \\ cAsBcC + sAsC & sAsBcC - cAsC & cBcC & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & -X \\ 0 & 1 & 0 & -Y \\ 0 & 0 & 1 & -Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5.1) $$

5.1.3 Ethernet Communication with Control Computer

Communication of position feedback and motion commands with the robot was attained by means of a KUKA service application which makes interaction with external systems possible through an Ethernet interface [54]. The service consists of a message framework and an application programming interface (API) that enables programs running on the robot controller (written in KRL programming language) to act as clients and connect to remote server applications.

Two types of Ethernet connections are allowed: TCP and UDP. Both use the same message structure. Each message is 332 bytes long and is divided into six sections:

- a section that encodes the version of the service application installed (1 byte long)
- a section of seven one-byte character values that is reserved for future use (7 bytes long)
- a section of twenty eight 32-bit integers (112 bytes long)
- a section of thirty two 32-bit floating point numbers (128 bytes long)
- a section of eighty one one-byte character values (81 bytes long)
- a padding section of three unused bytes

The API through which KRL programs interact with Ethernet-KRL consists of five function calls:

- INT ETH OPEN(IPADDRESS[]: OUT, PORTID: IN, TYPE: IN, CONNECTIONID: OUT), which establishes a connection with the host
- INT ETH CLOSE(CONNECTIONID: IN), which closes an established connection
- INT ETH WRITE(CONNECTIONID: IN, IPARMS[]: OUT, RPARMS[]: OUT, CPARMS[]: OUT), which sends a request to the host

- INT ETH READ(CONNECTIONID: IN, IPARMS[]: OUT, RPARMS[]: OUT, CPARMS[]: OUT, TIMEOUT: IN), which reads a response received from the host

- INT ETH RESET(), which closes all unattended connections

In addition to the message structures, an application level protocol is required to achieve meaningful communication between the robot controller and the main control computer. An example server application supplied with the Ethernet-KRL software defines a protocol to meet this requirement [54]. To comply with this protocol, the first element of the array of integers in the message structure contains a control code. There are several possible control codes, the most important of which correspond to the commands CONNECT, DISCONNECT, START, STOP and GO HOME. Robot programs periodically transmit to the server TCP requests that warrant responses containing any pending motion commands. The control computer responds by sending such motion commands in messages with the START control code. Each motion command message also contains a repetition number and a length value which are mapped, respectively, to an integer element and a floating point element in the corresponding arrays in the message structure. These commands are subsequently carried out by the robot, which then sends an acknowledgement message. The robot controller also periodically transmits UDP messages containing position feedback as well as robot status information such as active tool and base. For the purpose of the seam tracking application, the server application response structure was adjusted to allow general motions in 6-degrees of freedom by adding an array of six floating point values instead of the single length parameter to the operation control message. A structure of type FRAME, which consists of three length values in mm corresponding to translations along the three Cartesian axes and three rotation angles in degrees, was added to the OP CONTROL message structure

at the client-side KRL program. The code changes to the KRL client program and the C++ server application are shown in appendix A section A.3.

5.2 Vision System

The vision system consisted of a Basler Scout scA1000-30fm monochrome CCD camera, fitted with a Myutron FV-7583 lens. The control computer was equipped with a FireWire PCI card to which the camera was connected. Interfacing of the camera to the control computer was attained by means of Pylon driver, which comprises an application programming interface (API) and a viewer software which facilitated manual configuration and image acquisition. Figure 5.6 on page 56 shows a photograph of the camera and lens mounted to the mechanical interface bracket.

5.3 Camera Calibration

Calibration of the camera by obtaining the parameters of the transformation that describes its mapping of points from the scene into images is necessary for the inverse process of mapping points from images into ones in the robot environment. This section describes the calibration process that was followed in this project.

5.3.1 Determination of Focal Length, Pixel Dimension and Focal Point Distance

Camera calibration is a problem of many degrees of freedom. By constraining the scene geometry to a single plane that is parallel to the camera's principal plane, a smaller part of the calibration problem may be isolated and approached separately. This part is determination of the focal length and pixel dimension combination and the distance of the focal point from the camera's principal plane. This may be illustrated by considering an imaging scenario in a two-dimensional world (figure 5.7). As it can be seen in figure 5.7, the imaged line length, x, is given in pixels by:

$$\frac{x/K_c}{f} = \frac{X}{z - z_0} \tag{5.2}$$

where K_c is the number of pixels per unit length. This expression may be made linear in the distance from the camera z by rearranging it as:

$$\frac{1}{x} = \frac{1}{fK_cX}z - \frac{1}{fK_cX}z_0 \tag{5.3}$$

From this, the parameters K_cf and z_0 can be determined by taking several images of a known dimension on a plane parallel to the camera's principal plane. The points can then be plotted on a graph whose ordinate is the reciprocal of the imaged length and whose abscissa is the distance along the camera's axis to a fixed point (z in equation 5.3). The slope and intercept of the line can then be equated to the expressions in equation 5.3 above and solved for K_cf and z_0.

Circle Diameter and Centre

In performing the procedure described above for partial calibration of the camera, errors in determining the actual length and the imaged length contribute to

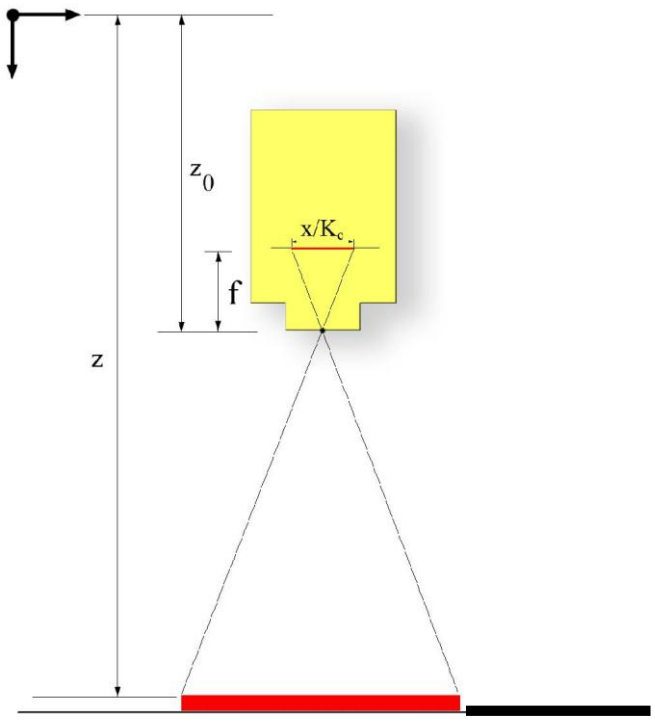


Figure 5.7: Schematic of Test for Determination of Focal Length and Pixel Dimension

errors in the resulting calibration parameters. It is therefore necessary to use a robust dimension in performing the procedure. One possible robust dimension is the diameter of a circle determined from the coordinates of several points in its perimeter. An additional advantage of this measure is that the eccentricity of the imaged circle can be used to indicate the error in parallelism of the scene plane and the camera's principal plane.

By specifying the coordinates of three points in general position, it is possible to determine the equation of a unique circle that passes through them. However, the resulting circle equation and consequently the diameter and centre coordinates obtained therefrom will not be robust, and any errors in the supplied

perimeter-point coordinates will cause errors in the obtained circle parameters. An alternative method is to use the coordinates of more points on the perimeter than is required to determine the circle. Errors in the supplied coordinates will mean that it is not possible to find a circle that passes exactly through all the points. However, it is always possible to find a circle that fits the points in a least-squares sense. This circle, and consequently its diameter and centre coordinates, is robust to errors in the coordinates of the points that were used to obtain it.

The same argument can be used to justify using more than five points, which is the minimum number required, to specify the elliptical image of a circle in the scene. It is appropriate to note here that if the requirement of the scene plane being exactly parallel to the camera's principal plane is met, the image of a circle in the scene will be a circle. However, exact parallelism between the two planes cannot be achieved because of errors in the camera position feedback values and the measurement of the scene plane. It is therefore reasonable to fit a general ellipse in the image rather than a circle.

To explain the procedure for determining an ellipse (or a circle) that passes through a number of points, a good starting point would be examining the algebraic description of an ellipse. A possible formulation of such description is to state that an ellipse is a collection of points with coordinates (x,y) that satisfy the equation:

$$\frac{(x\cos(\theta) + y\sin(\theta) - x_0)^2}{a^2} + \frac{(x\sin(\theta) - y\cos(\theta) - y_0)^2}{b^2} - R^2 = 0 \qquad (5.4)$$

The left-hand side of the equation above can be written as a quadratic form of the homogeneous representation of the point $[x, y]'$ given by $[x, y, 1]'$. The

equation can then be written as:

$$x^T C x = 0 \tag{5.5}$$

C here is a 3 × 3 quadratic form constructed from the parameters x_0, y_0, a, b, θ and R from equation 5.4 above and x is the homogeneous representation of a point in the ellipse. This last equation is quadratic in the coordinates of x, but it is linear in the elements of C. It is therefore possible to find a least-squares solution for the elements of C that satisfies equation 5.4 for a set of points x_i. Each point from the set gives a linear equation in the coordinates of C. To illustrate, if C is given by:

$$C = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{12} & c_{22} & c_{23} \\ c_{13} & c_{23} & c_{33} \end{bmatrix} \tag{5.6}$$

Then a point $x_i = [\, x_i \; y_i \; 1 \,]^T$ on the ellipse gives the equation:

$$c_{11}x^2_i + c_{22}y^2_i + c_{33} + 2c_{12}x_iy_i + 2c_{13}x_i + 2c_{23}y_i = 0 \tag{5.7}$$

n points on an ellipse then result in the matrix equation:

$$\begin{bmatrix} x^2_1 & y^2_1 & 1 & 2x_1y_1 & 2x_1 & 2y_1 \\ x^2_2 & y^2_2 & 1 & 2x_2y_2 & 2x_2 & 2y_2 \\ \dots & \dots & \dots & \dots & \dots & \dots \\ x^2_n & y^2_n & 1 & 2x_ny_n & 2x_n & 2y_n \end{bmatrix} \begin{bmatrix} c_{11} \\ c_{22} \\ c_{33} \\ c_{12} \\ c_{13} \\ c_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \dots \\ 0 \end{bmatrix} \tag{5.8}$$

The singular value decomposition of the $n \times 6$ matrix above always exists, and the right singular vector corresponding to the smallest singular value is the best solution for the ellipse parameters c_{ii}. It was necessary to avoid numerical errors by shifting and scaling the points such that they become centred at the origin with a standard deviation of $\sqrt{2}$ as recommended by Hartley and Zisserman [45].

Once the quadratic form C representing the imaged circle was found, it was transformed to a conic centred at the origin to extract the properties of the ellipse it represents. The transformation which maps C to a an ellipse centred at the origin is a translation:

$$H = \left[\begin{array}{c|c} I_2 & t_{2\times 1} \\ \hline 0^T_{2\times 1} & 1 \end{array}\right]$$

An ellipse centred at the origin has the form:

$$C_0 = \left[\begin{array}{c|c} A_{2x2} & 0_{2x1} \\ \hline 0_{2}^{T}{}_{x\,1} & -r^2 \end{array}\right]$$

The translated ellipse is then of the form:

$$C = H^T C_0 H = \left[\begin{array}{c|c} A_{2x2} & A_{2x2}t_{2x1} \\ \hline t_{T2x1}A_{2x2} & t^T A t - r_2 \end{array}\right]$$

Therefore, by comparing this partitioned expression of C to the expression in equation 5.6 we have:

$$A = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \tag{5.9}$$

$$t = A^{-1} \begin{bmatrix} c_{13} \\ c_{23} \end{bmatrix} \tag{5.10}$$

$$r = \sqrt{t^T A t - c_{33}} \tag{5.11}$$

The major and minor diameters of the ellipse can then be obtained from r and the Eigenvalues of A, which are real valued since A is symmetric. If λ_1 and λ_2 are respectively the larger and smaller Eigenvalues of A, then the major and minor diameters of the ellipse are:

$$d_{maj} = 2r\sqrt{\lambda_2} \quad ; \quad d_{min} = 2r\sqrt{\lambda_1} \tag{5.12}$$

5.3.2 External Parameters and Principal Image Point Coordinates

Approximation of the model for the transformation between scene and image points is completed by determining the camera's focal length and pixel dimension described in section 5.3.1. This is because the design of the bracket and the dimensions of the camera are known and can be used to compute the transformation between the camera coordinate frame and the robot flange coordinate frame. This transformation can be post multiplied by the matrix that transforms world coordinate frame points to the flange coordinate frame to relate points in the scene to the camera's coordinate frame. Finally, the focal length can be used to compute the transformation that maps points in the camera coordinate frame to image points if the camera axis is assumed to pass through the centre of the image.

However, it is unlikely that there will be no errors in any of these transformations and parameters. The calibration process must therefore be refined by estimating the following parameters of the camera:

- the focal length f in *mm* and pixel side length reciprocal K_c in *pixel/mm* (square pixels were assumed)

- the six parameters which determined the transformation from the robot end effector coordinate frame to the camera coordinate frame (three for translation and three for rotation)

- the horizontal and vertical offsets between the centre of the camera coordinate system and the pixel with coordinates (0, 0) in the image x_{P0} and y_{P0}

The principles of projective geometry show that only nine of the previous ten parameters are independent. This is because the transformation from a scene point to a camera point is represented by a 3 × 4 projective matrix with 11 independent entries up to scale, and two of the entries are eliminated by assuming square pixels.

Calibration is essentially an optimization problem. The parameters being estimated must be assigned values which minimize an error function, which is an observed deviation from the assumed model. A description of selected error functions and used optimization methods will follow.

Calculation of Error

Multiple images were taken of a static scene, and coordinates of several image points were determined on the images on which they appear. The positions and orientations of the robot end effector were recorded when the images were taken. To calculate the error for a given set of parameter estimates, those estimates were used to calculate the parameters of the lines along which the rays travel in space. Each image point can be mapped to a ray in space through the scene point and the position of the camera centre at the time when that image was taken. For each pair of images, two such rays corresponding to the same scene point will intersect at the scene point in the absence of errors. In reality, however, the ray pair will not intersect due to errors in the camera parameter estimates. The sum of absolute distance values between pairs of image point rays was used as the estimation error to be minimized in the calibration process. The following methods were used for minimizing the objective function, and thereby estimating the parameters.

To derive a formula for calculating the error corresponding to a set of parameter estimates, a scenario was considered where two images are taken of some point in space. Let ${}^{1}T_2$ be the matrix for transforming points to the coordinate frame of the first image I_1 from the coordinate frame of the second image I_2. Furthermore, let the parameters for the transformation from the robot coordinate frame to the camera coordinate frame be the yaw, pitch, and roll angles A, B and C and the translation in the three coordinate axes, X , Y and Z. Additional parameters are the focal length f, the pixel side length K_c and the coordinates of the pixel corresponding to the origin of the camera coordinate frame x_{p0} and y_{p0}. The coordinates of scene point in the two images are $[x_{i1}, y_{i1}]$ and $[x_{i2}, y_{i2}]$.

The direction of the ray forming the point image on I_1 given by the vector:

$$r_1 = \begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} = \frac{1}{K_c} \begin{bmatrix} x_{i1} - x_{p0} \\ y_{i1} - y_{p0} \\ 0 \end{bmatrix} = \begin{bmatrix} \frac{x_{i1} - x_{p0}}{K_c} \\ \frac{y_{i1} - y_{p0}}{K_c} \\ f \end{bmatrix} \tag{5.13}$$

Similarly, the direction of the ray forming the point image on I_2 on the coordinate frame of I_1 is:

$$r_2 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} {}^{r}T_{-1}\,{}^{1}T_2\,{}^{r}T_c \begin{bmatrix} \frac{x_{i2} - x_{p0}}{K_c} \\ \frac{y_{i2} - y_{p0}}{K_c} \\ f \\ 1 \end{bmatrix} \tag{5.14}$$

If the two rays intersect in space, the point of intersection is the point in the scene corresponding to the two images. In reality, however, the rays often don’t

intersect as a result of noise and, more importantly, errors in the parameter estimates. In this case, the shortest vector connecting the two rays can be thought of as a calibration error vector. This vector can be obtained by taking a vector connecting any point in one ray to any point on the other and calculating its projection on a unit vector which is orthogonal to both the rays. The unit vector orthogonal to both the rays can be obtained by taking the cross product of vectors in the directions of the two rays and normalizing it.

$$\hat{n} = \frac{\|r_1 \times r_2\|}{r_1 \times r_2} \tag{5.15}$$

The vector connecting the two rays can be chosen as the vector that connects the focal points of the camera in the two positions. The error calculated in this way is therefore:

$$e = \hat{n}^T \left(\begin{bmatrix} 0 \\ 0 \\ f \end{bmatrix} - \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} {}^{r}T_{c}^{-1} {}^{1}T_2 {}^{r}T_c \begin{bmatrix} 0 \\ 0 \\ f \\ 1 \end{bmatrix} \right) \tag{5.16}$$

These geometric relationships are described in figure 5.8.

Nonlinear Least Squares Problem Formulation

Nonlinear least squares adjustment was the methods used to address the calibration problem. By taking combinations of point matches from different images, a set of error values could be calculated for each set of parameter estimates. The

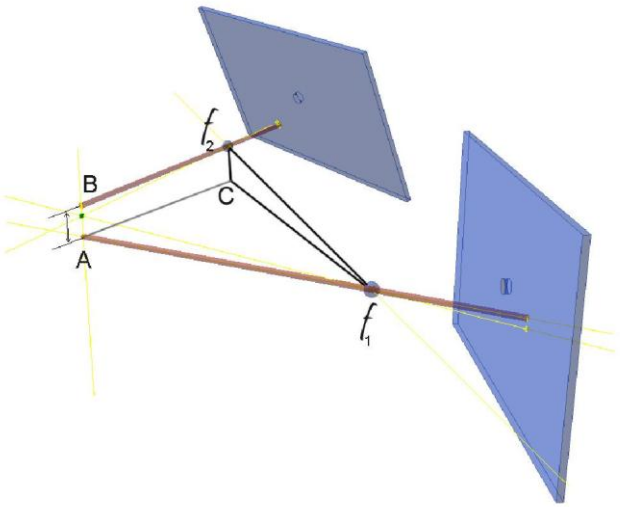


Figure 5.8: Ray Intersection Error

function that maps a set of parameter estimates to a set of corresponding errors given by:

$$\begin{bmatrix} \tilde{e}_1 \\ \tilde{e}_2 \\ \cdots \\ \tilde{e}_3 \end{bmatrix} = \psi \left(\begin{bmatrix} f \\ K_c \\ X \\ Y \\ Z \\ A \\ B \\ C \\ x_{P0} \\ y_{P0} \end{bmatrix} \right) \tag{5.17}$$

is a nonlinear function, and it must be linearized before the least squares approximation can be performed. Linearization is performed by taking the first two terms of the Taylor series representation of the function. This is given by:

$$\begin{bmatrix} e_1 \\ e_2 \\ \cdots \\ e_3 \end{bmatrix} = \psi\left(\begin{bmatrix} f_0 \\ K_{c0} \\ X_0 \\ Y_0 \\ Z_0 \\ A_0 \\ B_0 \\ C_0 \\ x_{P_i 0} \\ y_{P_i 0} \end{bmatrix}\right) + \begin{bmatrix} \frac{\partial e_1}{\partial f} & \frac{\partial e_1}{\partial K_e} & \frac{\partial e_1}{\partial X} & \frac{\partial e_1}{\partial Y} & \frac{\partial e_1}{\partial Z} & \frac{\partial e_1}{\partial A} & \frac{\partial e_1}{\partial B} & \frac{\partial e_1}{\partial C} & \frac{\partial e_1}{\partial x_{P_i}} & \frac{\partial e_1}{\partial y_{P_i}} \\ \frac{\partial e_2}{\partial f} & \frac{\partial e_2}{\partial K_e} & \frac{\partial e_2}{\partial X} & \frac{\partial e_2}{\partial Y} & \frac{\partial e_2}{\partial Z} & \frac{\partial e_2}{\partial A} & \frac{\partial e_2}{\partial B} & \frac{\partial e_2}{\partial C} & \frac{\partial e_2}{\partial x_{P_i}} & \frac{\partial e_2}{\partial y_{P_i}} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \frac{\partial e_m}{\partial f} & \frac{\partial e_m}{\partial K_e} & \frac{\partial e_m}{\partial X} & \frac{\partial e_m}{\partial Y} & \frac{\partial e_m}{\partial Z} & \frac{\partial e_m}{\partial A} & \frac{\partial e_m}{\partial B} & \frac{\partial e_m}{\partial C} & \frac{\partial e_m}{\partial x_{P_i}} & \frac{\partial e_m}{\partial y_{P_i}} \end{bmatrix} \begin{bmatrix} \Delta f \\ \Delta K_c \\ \Delta X \\ \Delta Y \\ \Delta Z \\ \Delta A \\ \Delta B \\ \Delta C \\ \Delta x_{P_i} \\ \Delta y_{P_i} \end{bmatrix} \tag{5.18}$$

The matrix of partial derivatives in the equation above is known as the Jacobian matrix. In this problem, it is populated by differentiating the expression for the error in equation 5.16 (page 69). Let t be any of the transformation parameters to be estimated. Differentiating equation 5.16 in terms of t gives:

$$\frac{\partial e_h}{\partial t} = \frac{\partial \hat{n}}{\partial t}^T \left[I_{3x3} \; 0_{3x1} \right] \prod_{i=1}^{j} T_i f + \hat{n}^T \frac{\partial}{\partial t} \left[I_{3x3} \; 0_{3x1} \right] \prod_{i=1}^{j} T_i f + \hat{n}^T \left[I_{3x3} \; 0_{3x1} \right] \sum_{i=1}^{j} \left(\frac{\partial T_i}{\partial t} \prod_{\substack{k=1 \\ k \neq i}}^{j} T_k \right) f \tag{5.19}$$

where:

$$\frac{\partial \hat{n}}{\partial t} = \frac{\partial}{\partial t} \left(\frac{r_1 \times r_2}{\| r_1 \times r_2 \|} \right) = \frac{\| r_1 \times r_2 \| \frac{\partial}{\partial t}(r_1 \times r_2) - (r_1 \times r_2) \frac{\partial}{\partial t} \| r_1 \times r_2 \|}{\| r_1 \times r_2 \|^2} \tag{5.20}$$

$$\frac{d}{dt}(r_1 \times r_2) = \left(\frac{d}{dt} r_1 \times r_2\right) + \left(r_1 \times \frac{d}{dt} r_2\right) \tag{5.21}$$

and

$$\frac{d}{dt}\|r_1 \times r_2\| = \frac{\left(r_1 \times r_2, \frac{\partial}{\partial t}(r_1 \times r_2)\right)}{\|r_1 \times r_2\|} \tag{5.22}$$

where $(\cdot,\cdot)$ denotes the inner product. To obtain the derivatives for the specific parameters, the terms in the formulae above are evaluated and substituted. These are given by:

$$\frac{\partial r_1}{\partial f} = \begin{bmatrix} \frac{x_{i1} - x_{p0}}{K} \\ -\frac{y_{i1} - y_{p0}}{K} \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{5.23}$$

$$\frac{\partial r_1}{\partial x_{p0}} = \frac{\partial}{\partial x_{p0}} \begin{bmatrix} \frac{x_{i1} - x_{p0}}{K} \\ -\frac{y_{i1} - y_{p0}}{K} \\ f \end{bmatrix} = \begin{bmatrix} \frac{1}{K} \\ 0 \\ 0 \end{bmatrix} \tag{5.24}$$

$$\frac{\partial r_1}{\partial y_{p0}} = \frac{\partial}{\partial y_{p0}} \begin{bmatrix} \frac{x_{i1} - x_{p0}}{K} \\ -\frac{y_{i1} - y_{p0}}{K} \\ f \end{bmatrix} = \begin{bmatrix} 0 \\ \frac{1}{K} \\ 0 \end{bmatrix} \tag{5.25}$$

$$\frac{\partial r_1}{\partial K} = \frac{\partial}{\partial K} \begin{bmatrix} \frac{x_{i1} - x_{p0}}{K} \\ -\frac{y_{i1} - y_{p0}}{K} \\ f \end{bmatrix} = \begin{bmatrix} -\frac{x_{i1} - x_{p0}}{K^2} \\ \frac{y_{i1} - y_{p0}}{K^2} \\ 0 \end{bmatrix} \tag{5.26}$$

r_1 is not affected by the camera position parameters, therefore:

$$\frac{\partial r_1}{\partial X} \quad \frac{\partial r_1}{\partial Y} \quad \frac{\partial r_1}{\partial Z} \quad \frac{\partial r_1}{\partial A} \quad \frac{\partial r_1}{\partial B} \quad \frac{\partial r_1}{\partial C} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.27}$$

Similarly,

$$\frac{\partial r_2}{\partial f} = {}^{i}T_j \frac{\partial}{\partial f} \begin{pmatrix} \begin{bmatrix} \frac{x_{i2}-x_{p0}}{K_x} \\ \frac{y_{i2}-y_{p0}}{K_y} \\ f \end{bmatrix} \\ \begin{bmatrix} 0 \end{bmatrix} \end{pmatrix} = {}^{i}T_j \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \tag{5.28}$$

$$\frac{\partial r_2}{\partial x_{p0}} = {}^{i}T_j \frac{\partial}{\partial x_{p0}} \begin{pmatrix} \begin{bmatrix} \frac{x_{i2}-x_{p0}}{K_x} \\ \frac{y_{i2}-y_{p0}}{K_y} \\ f \end{bmatrix} \\ \begin{bmatrix} 0 \end{bmatrix} \end{pmatrix} = {}^{i}T_j \begin{bmatrix} \frac{1}{K_x} \\ 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.29}$$

$$\frac{\partial r_2}{\partial y_{p0}} = {}^{i}T_j \frac{\partial}{\partial y_{p0}} \begin{pmatrix} \begin{bmatrix} \frac{x_{i2}-x_{p0}}{K_x} \\ \frac{y_{i2}-y_{p0}}{K_y} \\ f \end{bmatrix} \\ \begin{bmatrix} 0 \end{bmatrix} \end{pmatrix} = {}^{i}T_j \begin{bmatrix} 0 \\ \frac{1}{K_y} \\ 0 \\ 0 \end{bmatrix} \tag{5.30}$$

$$\frac{\partial r_2}{\partial K_c} = {}^{i}T_j \frac{\partial}{\partial K_c} \begin{pmatrix} \begin{bmatrix} \frac{x_{i2}-x_{p0}}{K_x} \\ \frac{y_{i2}-y_{p0}}{K_y} \\ f \end{bmatrix} \\ \begin{bmatrix} 0 \end{bmatrix} \end{pmatrix} = {}^{i}T_j \begin{bmatrix} \frac{x_{i2}-x_{p0}}{K_x^2} \\ \frac{y_{i2}-y_{p0}}{K_y^2} \\ 0 \\ 0 \end{bmatrix} \tag{5.31}$$

$$\frac{\partial r_t}{\partial X} = \frac{\partial}{\partial X}\,{}^{i}T_j \begin{bmatrix} \frac{x_{i2}-x_{p0}}{K} \\ \frac{y_{i2}-y_{p0}}{K} \\ f \\ 0 \end{bmatrix} \tag{5.32}$$

$$\frac{\partial r_t}{\partial Y} = \frac{\partial}{\partial Y}\,{}^{i}T_j \begin{bmatrix} \frac{x_{i2}-x_{p0}}{K} \\ \frac{y_{i2}-y_{p0}}{K} \\ f \\ 0 \end{bmatrix} \tag{5.33}$$

$$\frac{\partial r_t}{\partial Z} = \frac{\partial}{\partial Z}\,{}^{i}T_j \begin{bmatrix} \frac{x_{i2}-x_{p0}}{K} \\ \frac{y_{i2}-y_{p0}}{K} \\ f \\ 0 \end{bmatrix} \tag{5.34}$$

$$\frac{\partial r_t}{\partial A} \quad \frac{\partial r_t}{\partial B} \quad \frac{\partial r_t}{\partial C} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \tag{5.35}$$

$${}^{i}T_j = {}^{c}T_r^{ri}\,T_{rj}^{c}\,T^{-1}{}_{r} \tag{5.36}$$

so, since the derivative of an inverse matrix is given by:

$$\frac{\partial T^{-1}}{\partial t} = T^{-1}\,\frac{\partial T}{\partial t}\,T^{-1} \tag{5.37}$$

the derivative of the camera pose transformation matrix is:

$$\frac{\partial {}^{i}T_j}{\partial t} = \frac{\partial {}^{c}T_{ri}\,{}^{r}T_{rj}\,{}^{c}T^{-1}}{\partial t} = \frac{\partial {}^{c}T_r}{\partial t}\,{}^{r}T_{rj}\,{}^{c}T^{-1} + {}^{c}T_{rri}\,\frac{\partial {}^{r}T_{rj}}{\partial t}\,{}^{c}T^{-1} + {}^{c}T_{rri}\,{}^{r}T_{rj}\,\frac{\partial {}^{c}T^{-1}}{\partial t}$$

The transformation matrix from the robot end effector coordinate frame to the camera coordinate frame has the same form as that between the base and tool coordinate frames (equation 5.1). If we denote the translation part of the transformation by $T_{trans,}$ the derivatives of the transformation matrix are given by:

$$\frac{\partial T}{\partial A} = \begin{bmatrix} 0 & -sAcB & cAcB & 0 \\ 0 & -sAsBsC - cAcC & cAsBsC - sAcC & 0 \\ 0 & sAsBcC + cAsC & cAsBcC + sAsC & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} T_{trans} \qquad (5.39)$$

$$\frac{\partial T}{\partial B} = \begin{bmatrix} 0 & -cAsB & -sAsB & -cB \\ 0 & cAcBsC & sAcBsC & -sBsC \\ 0 & cAcBcC & sAcBcC & -sBcC \\ 0 & 0 & 0 & 0 \end{bmatrix} T_{trans} \qquad (5.40)$$

$$\frac{\partial T}{\partial C} = \begin{vmatrix} 0 & 0 & 0 & 0 \\ c_A s_B c_C + s_A s_C & s_A s_B c_C - c_A s_C & c_B c_C & 0 \\ c_A s_B s_C + s_A c_C & -s_A s_B s_C - c_A c_C & -c_B s_C & 0 \\ 0 & 0 & 0 & 0 \end{vmatrix} T_{trans} \quad (5.41)$$

$$\frac{\partial T}{\partial X} = \begin{bmatrix} 0\,0\,0 & -cAcB \\ 0\,0\,0 & -cAsBsC + sAcC \\ 0\,0\,0 & -cAsBcC - sAsC \\ 0\,0\,0 & 0 \end{bmatrix} \tag{5.42}$$

$$\frac{\partial T}{\partial Y} = \begin{bmatrix} 0\,0\,0 & -sAcB \\ 0\,0\,0 & -sAsBsC - cAcC \\ 0\,0\,0 & -sAsBcC + cAsC \\ 0\,0\,0 & 0 \end{bmatrix} \tag{5.43}$$

$$\frac{\partial T}{\partial Z} = \begin{bmatrix} 0\,0\,0 & sB \\ 0\,0\,0 & -cBsC \\ 0\,0\,0 & -cBcC \\ 0\,0\,0 & 0 \end{bmatrix} \tag{5.44}$$

This completes the formulae required to populate the Jacobian matrix for applying nonlinear least squares to the camera calibration problem. The Matlab code for evaluating the derivatives above at the required points and formulating the Jacobian matrix is in appendix A section A.1 (page 109).

The process of nonlinear least squares adjustment involves iteratively adjusting the parameter estimates by subtracting the product of the pseudo-inverse of the Jacobian by the errors vector. Two methods were used for obtaining the pseudo-inverse of the Jacobian. The first was by means of using the standard pseudo-inverse function of Matlab, and the other was using singular value decompositoin and setting the singularity threshold to a larger value than the standard limit. Matlab code written for the nonlinear adjustment procedure is contained in appendix A subsection A.1.1 (page 115).

5.4 Seam and Feature Detection

In order to track the seam in multiple images, it is necessary to:

- identify the pixels corresponding to the seam in each of the images, and
- identify the position of the seam in space as a function of a sampling parameter, t

Two methods were tested for detecting the seam in individual images. The first method was using linear filtering of the image and applying a Canny edge detection algorithm with non-maximal supression. The second method was an adaptation of Harris corner detector introduced by Harris [56]. It is based on using the Eigenvalues of the Jacobian of the image function. Since the images supplied by the camera are discrete, the Jacobian J was approximated with a finite difference operator convolved with a gaussian kernel. Points where the larger of the two Eigenvalues are highest correspond to curves in the image across which the rate of change is the highest. This is often true of the seam pixels.

The smaller of the two Eigenvalues of the Jacobian at a point is a measure of the smallest rate of change at that pixel along any direction. Points where the smaller Eigenvalues are highest, therefore, are often used as keypoints in the images. Such feature points can be matched across different images and triangulated to obtain positions of the corresponding points in the scene. After linear filtering the image with the finite difference and Gaussian kernels, a threshold is applied to find the possible feature points. For each of the detected points, a local search is performed and only a single pixel point in a neighbourhood is selected as the feature. The code for the feature detection program is in

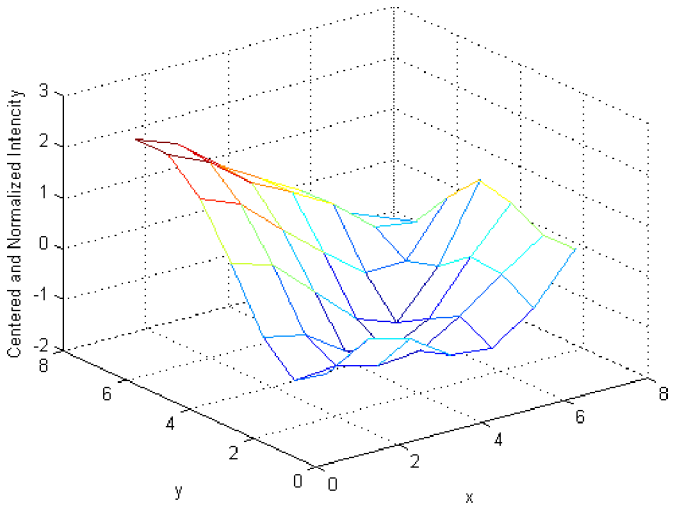


Figure 5.9: Neighbourhood Descriptor for Feature

Tracking Appendix A section A.4.

Feature point on the seam are especially useful, as they are likely to correspond to three dimensional points on the seam. After general feature points are found, the seam features are detected using the same method as general feature points with the exception that the search is restricted to seam points.

To track the detected features, a neighbourhood of 7 × 7 pixels around the detected keypoint position in the first image was sampled. The 49 values were then scaled and normalized such that they had a zero-mean and a standard deviation of one. This neighbourhood descriptor was then compared to similar ones on the following images, with varying center position. The center position corresponding to the smallest sum of square error in the neighbourhood intensities was chosen. This process is described by Szeliski [53]. In this project, the search was constrained to a five-pixel strip around the calculated epipolar line. Figure 5.9 shows a sample neighbourhood descriptor.

5.5 Bundle Adjustment

In calculating positions of scene points from multiple images by triangulation, errors in robot feedback positions are propagated directly to the calculated scene geometry. To address this problem, bundle adjustment was used to obtain accurate scene point coordinates as well as corrections to the robot position and pose. An detailed review of bundle adjustment techniques was given by Triggs et al. [55].

Each image feature point was used to formulate a set of equations that relate the coordinates of the image point to those which would be obtained given the current estimates of camera internal and external calibration parameters and the scene point coordinates. This process of calculating approximate feature image coordinates by applying the estimated camera model to estimated scene geometry is called reprojection. After formulating reprojection error equations for several images, they were iteratively minimized from the starting estimates to obtain the final values for scene geometry as well as corrected camera poses.

An alternative, though equivalent, formulation for geometric image formation was used for reprojection [45]. The coordinate system of the camera is placed at its projection centre with the z-axis along the camera axis. This way, only the first three homogeneous coordinates of a scene point are required to determine its corresponding image point position, and scale is insignificant. The x- and y-axes are placed along those of the image. The image $\mathbf{x}$ of a point ${}^{\{C\}}\mathbf{X}$ in this coordinate frame is given by:

$$x = \begin{bmatrix} -K_c f & 0 & p_{x0} \\ 0 & -K_c f & p_{y0} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3\times3} & 0_{3\times1} \end{bmatrix} \sim C \sim X$$

$$\Rightarrow h = K \begin{bmatrix} I & 0 \end{bmatrix} C X \tag{5.45}$$

The upper triangular matrix K in this expression encodes the internal parameters of the camera.

Because, in general, scene points are not expressed in terms of this camera coordinate frame beforehand, they must be transformed such that they are. This is obtained by applying a Euclidian transformation, which comprises a rotation R and a translation t:

$$CX = \begin{bmatrix} R & t \\ 0^T_{3\times1} & 1 \end{bmatrix} X \tag{5.46}$$

These rotation matrix and translation vector encode the external parameters of the camera, and the point X, trivially, encodes the position of the scene point. The complete reprojection of the point X then becomes:

$$\Rightarrow hx = K R \begin{bmatrix} t \end{bmatrix} X \tag{5.47}$$

Reprojection error is the difference between the actual image point and this reprojected point. In this project, the error was taken to be the difference between the image point coordinates and the non-homogeneous reprojected point

coordinates.

$$e = x_m - \left(xp\left(\frac{1}{\left| \begin{matrix} h & 0 & 0 & 1 \end{matrix} \right| \begin{matrix} i & t & X \end{matrix}} \begin{matrix} h \\ \end{matrix} \right) \right) \left| \begin{matrix} i & 0 & K & R \end{matrix} \right| \begin{matrix} i \\ t \end{matrix} X \quad (5.48)$$

A column vector of reprojection errors of several points from several views is constructed. Each point contributes to the vector two entries corresponding to the x- and y-coordinate reprojection errors. The sum of squares of the entries of the reprojection error vector is iteratively minimized. To do this, a Jacobian matrix is formed with respect to the estimated scene point coordinates and camera internal and external calibration parameters. For each parameter u, the corresponding two entries of the Jacobian matrix corresponding to a specific scene point and a specific view are given by:

$$\frac{\partial e}{\partial u} \quad \frac{- \begin{matrix} h & 0 & 0 & 1 \end{matrix} \begin{matrix} i \\ x \end{matrix} \begin{matrix} h \\ \end{matrix} \left| \begin{matrix} 0 & \frac{\partial}{\partial u} \end{matrix} \right. \begin{matrix} h \\ i \end{matrix} \left. \begin{matrix} 0 & 0 & 1 & \frac{\partial}{\partial u} \end{matrix} \right| \begin{matrix} i \\ 2 \end{matrix} 0 \; x}{0 \quad 0 \quad 1 \quad x_2}$$

$$\left| \begin{matrix} h \\ \text{where } x = \text{K R} \end{matrix} \right| \begin{matrix} i \\ t \end{matrix} X, \text{ and}$$

$$\frac{\partial x}{\partial u} = \left| \frac{\partial}{\partial} \begin{matrix} h \\ \text{K} \end{matrix} \right| t \begin{matrix} i \\ X \end{matrix} + \text{K} \quad \text{h} \frac{\partial R}{\partial u} \left| \frac{\partial}{\partial} \begin{matrix} t \\ u \end{matrix} \right| \begin{matrix} i \\ \end{matrix} \quad X + \text{K} \begin{matrix} h \\ \text{R} \end{matrix} \quad \left| \frac{\partial X}{t \partial u} \right. \quad (5.50)$$

The code for computing reprojection error, populating the reprojection error Jacobian matrix, and carrying out iterative bundle adjustment is shown in appendix A, section A.5 (page 123).

5.6 Closure

This concludes the description of the experimental system and methods used in this project, summarized in figures 5.1 and 5.3 in the beginning of this chapter.

Chapter 6

Results

This chapter presents the results of the system calibration and qualification tests. Results of the calibration procedures are included in section 6.1. Sections 6.2 and 6.3 respectively present results of the seam and feature detection programs and the bundle adjustment implementation. The integrated seam tracking results are given in section 6.4.

6.1 Calibration Results

6.1.1 Focal Length

The first step in the calibration process was determination of the focal length of the lens, which was kept constant throughout the tests. The method followed was described in section 5.3.1 on page 60. Appendix B contains the images taken in this process in figure B.1 (page 134), and the robot feedback distances and the calculated major diameter values of the corresponding images in table B.1

(page 132). Figure 6.1 shows a plot of the feedback distance versus the reciprocal of the major radius, as well as a line fitted through the data. The equation of the line is:

$$\frac{1}{r_{maj}} = 1.098 \cdot 10^{-5} z - 0.008363$$

It was fitted by applying the method of least squares as described in section 4.5 on page 42 on the overdetermined linear system:

$$\begin{bmatrix} z_0 & 1 \\ z_1 & 1 \\ \vdots & \vdots \\ z_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} 1/r_{maj\,0} \\ 1/r_{maj\,1} \\ \vdots \\ 1/r_{maj\,n} \end{bmatrix}$$

The actual diameter of the scene circle was measured using a Vernier calliper to be 9.14 mm, and the height of its centre was measured by probing it using the robot and was found to be 761.66mm. From this, the focal length was calculated to be 69.9 mm by taking the pixel dimension to be 285 pixel/mm. The offset of the focal point from the camera's coordinate frame was calculated to be −87.36 mm.

6.1.2 External Parameters Calibration

This section contains the results of calibration of the remaining camera parameters through nonlinear least squares minimization of ray intersection errors as described in section 5.3.2 on page 66. Figure 6.2 shows the images used in the process, and table 6.1 shows the positions from which they were taken. As it

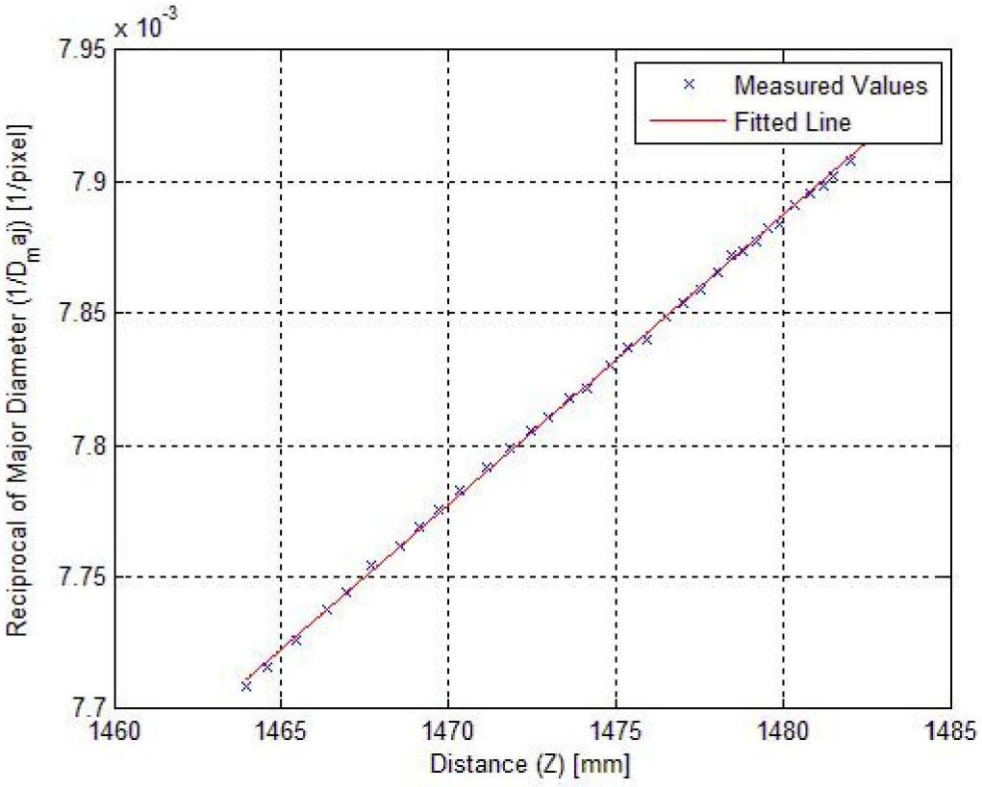
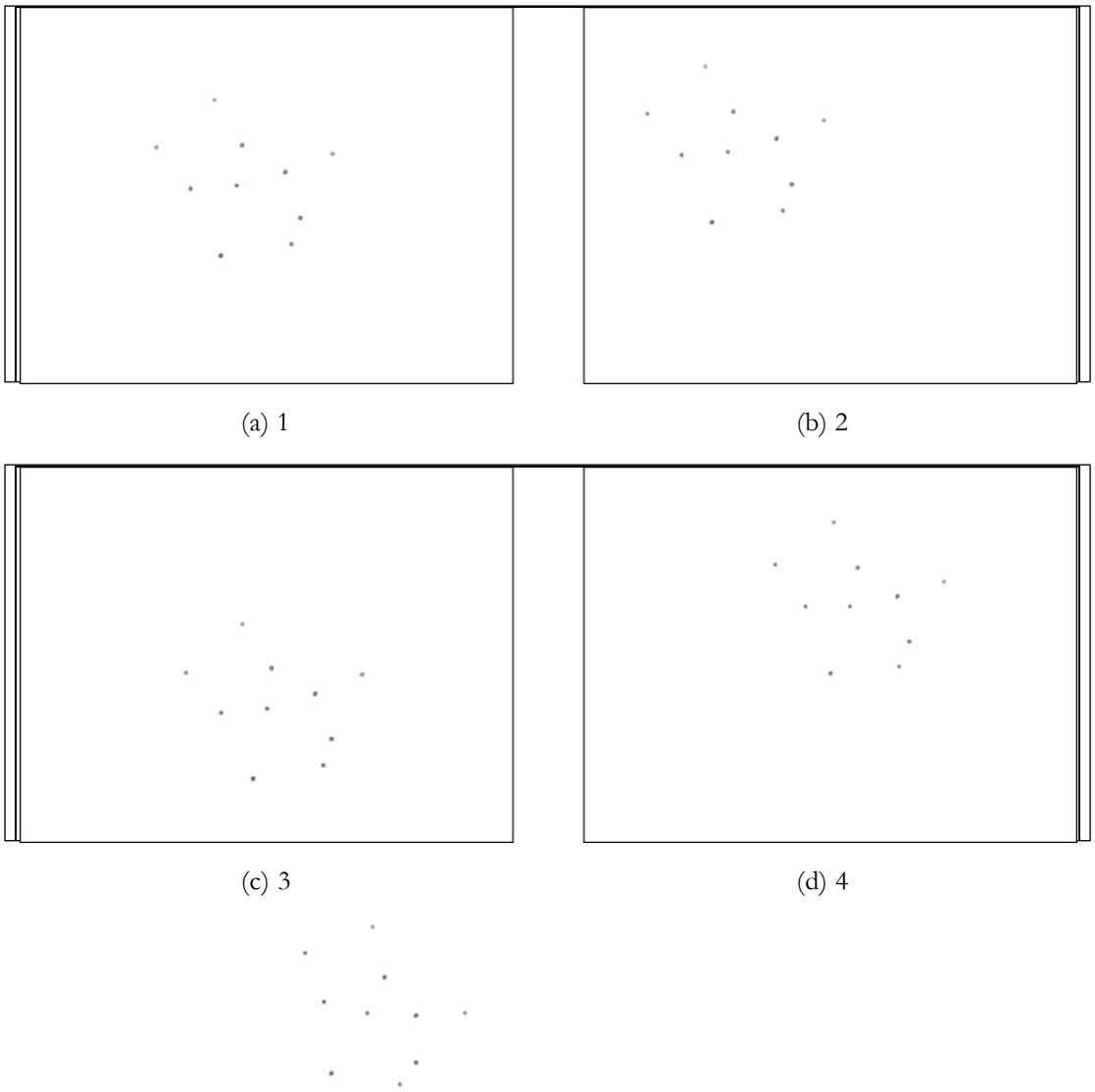


Figure 6.1: Focal Length Determination Test Results

can be seen in the images, the scene consisted of a set of ten points printed at random positions on a sheet of paper.

The ray intersection error is plotted against iterations in figure 6.3. The calibrated parameter values appear in table 6.2, and their time trends are plotted in figure 6.4 after being normalized such that $\sum_{i=0}^{n} x_i^2 = n$ with x_i denoting the parameter value at iteration i.



(a) 1 (b) 2

(c) 3 (d) 4

(e) 5

Figure 6.2: Images Used for Nonlinear Least Squares Calibration

Table 6. 1: Robot Positions For NLS Calibration Images

image	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
1	1179.53100	-79.757970	1473.60100	146.46720	-81.53249	-120.61680
2	1185.83100	-79.758100	1473.60400	146.46210	-81.53271	-120.61230
3	1173.89300	-68.818590	1473.61000	144.94890	-81.53227	-120.61390
4	1179.44100	-143.53410	1487.20000	176.48370	-82.55122	-147.02840
5	1235.52400	-198.42430	1493.45800	-173.7025	-83.82195	-142.24280

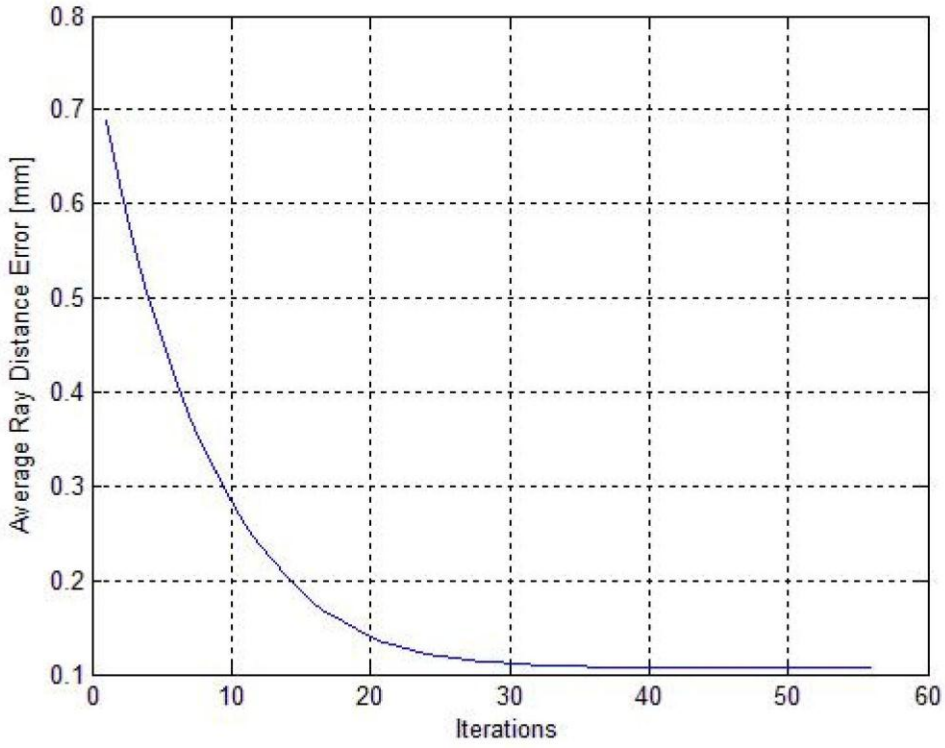


Figure 6.3: Nonlinear Least Squares using SVD: Calibration Results

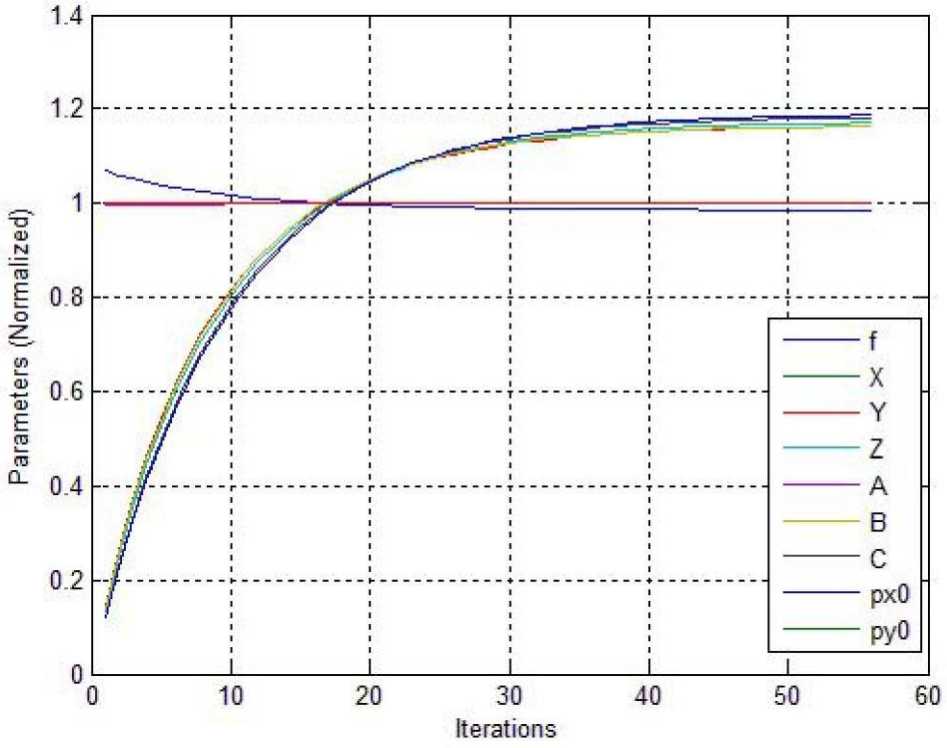
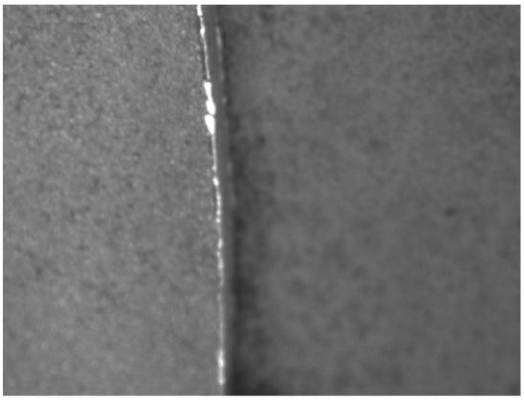


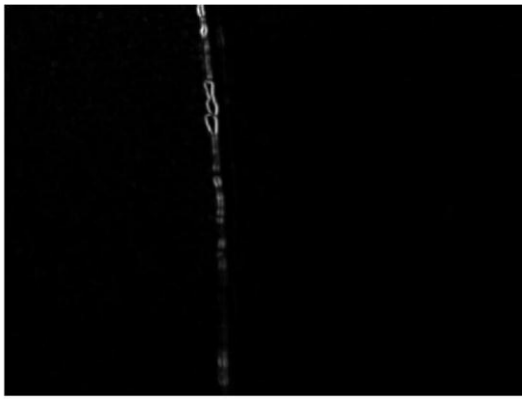
Figure 6.4: Nonlinear Least Squares using SVD Calibration: Normalized Parameter Values

Table 6.2: Nonlinear Least Squares using SVD: Parameter Values

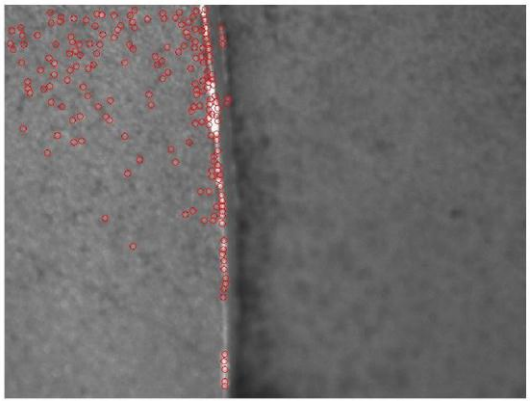
Parameter	Initial Value	Convergence Value
f[mm]	69.9	63.858
X[mm]	0.0	0.579
Y[mm]	0.0	0.595
Z[mm]	-87.36	-87.943
A[rad]	0.0	0.00782
B[rad]	0.0	0.00748
C[rad]	0.0	0.00244
p_{x0}[pixel]	517	517.00
p_{y0}[pixel]	389	389.00



(a) Raw Camera Image



(b) Seam Detection Result

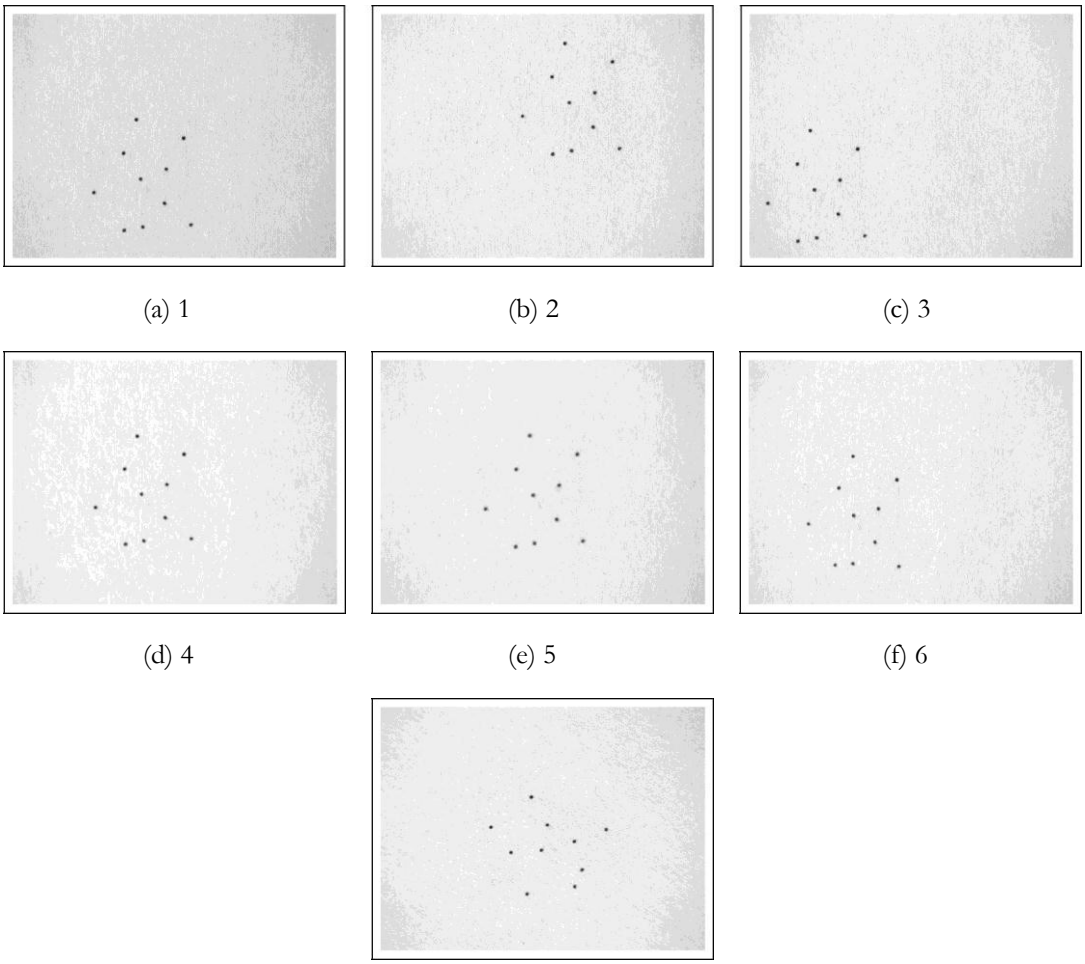


(c) Feature Detection Result

Figure 6.5: Seam and Feature Detection

6.2 Seam and Feature Detection

Figure 6.5(a) shows an image of the edge of the flange on the tank. The results of processing it with the two seam detection programs is shown in figure 6.5(b), and the result of processing it with the feature detection program is shown in figure 6.5(c).



(a) 1 (b) 2 (c) 3

(d) 4 (e) 5 (f) 6

(g) 7

Figure 6.6: Images Used for Testing Bundle Adjustment Implementation

6.3 Bundle Adjustment

The bundle adjustment implementation was tested with the images shown in figure 6.6. The images were taken by the camera mounted on the robot from different poses, which are given by the robot end effector frame parameters shown on table 6.3. As it is clear in the images, the scene contained ten points printed at random positions on a sheet of white paper. The actual placement of the points on the page are shown on figure 6.7.

Table 6.3: Robot Positions For Bundle Adjustment Testing

image	X [mm]	Y [mm]	Z [mm]	A [deg]	B [deg]	C [deg]
1	1472.2780	13.632430	1462.86200	179.220200	0.658427	-179.9413
2	1465.3150	4.7168490	1462.79000	179.220800	0.658153	-179.9411
3	1479.4930	14.891750	1462.94200	179.220800	0.658246	-179.9411
4	1474.0340	117.43210	1462.76200	179.222300	0.658014	-169.9894
5	1405.6950	114.85150	1442.06800	-179.69110	6.796363	-169.9193
6	1276.1050	-82.28652	1420.59100	-177.16890	20.21972	172.09360
7	1338.9800	-209.0233	1408.34300	122.499000	-10.8165	157.97300

Figure 6.7: Bundle Adjustment Test Image

Initialization values for the camera internal and external parameters were obtained from the outputs of the calibration programs. Scene geometry was initialized by triangulation of the points using a randomly selected pair of images.

Figure 6.8 shows a plot of reprojection error values during bundle adjustment. As shown in the figure, reprojection error was reduced by two orders of magnitude in a single bundle adjustment iteration. The process converged to a precision of 0.001 pixel in six iterations. Figure 6.9 shows the scene geometry with the optimal reprojection error, together with the initialization geometry used. At convergence, the average reprojection error was 1.68 pixel.

In addition to the scene geometry, bundle adjustment also produced corrections to the position of the camera centre and orientation of the camera at the

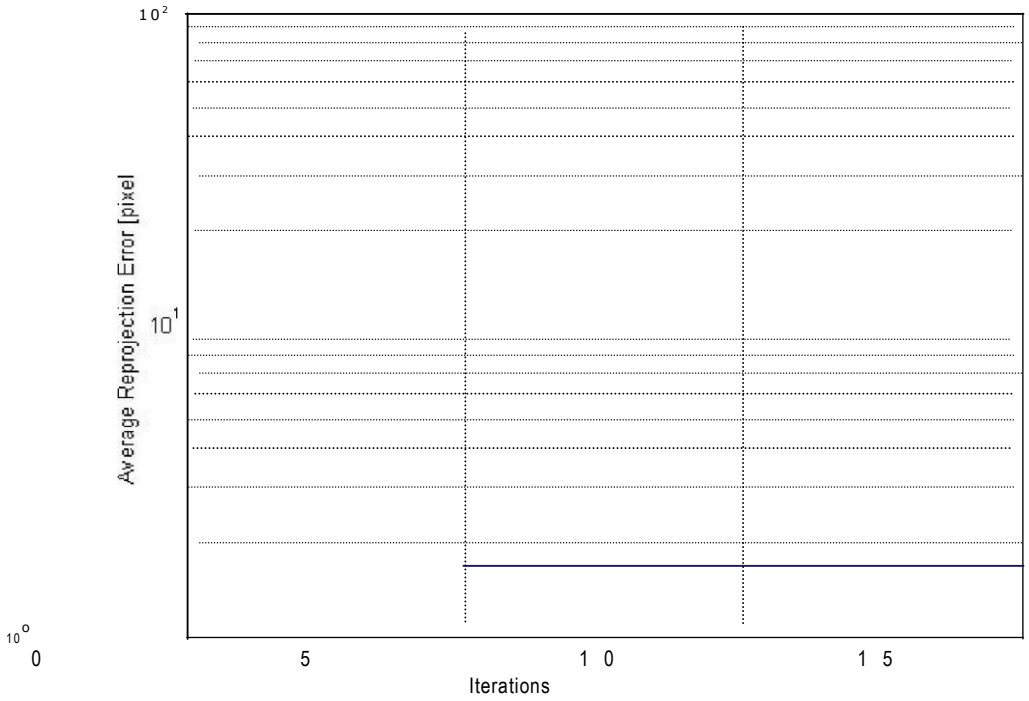


Figure 6.8: Reprojection Error during Bundle Adjustment

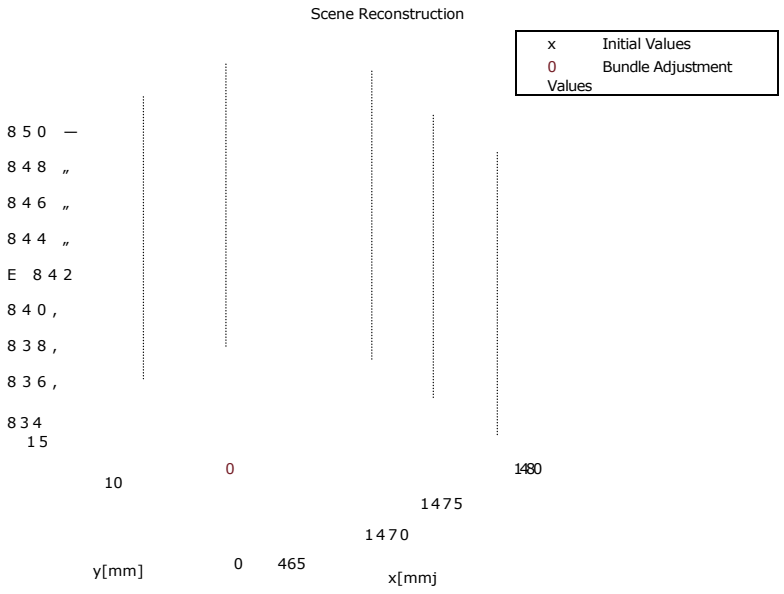


Figure 6.9: Scene Geometry: Initialization and Bundle Adjustment Result

Table 6.4: Bundle Adjustment Camera External Parameter Correction

image	*	X [mm]	Y [mm]	Z [mm]	A [rad]	B [rad]	C [rad]
1	i	-1479.1460	4.9529	-1543.6577	3.1458	0.0049	3.1432
	f	-1479.1405	4.9534	-1543.6535	3.1508	0.0050	3.1344
2	i	-1472.1380	-3.9534	-1543.6127	3.1458	0.0049	3.1432
	f	-1472.1331	-3.9532	-1543.6090	3.1509	0.0049	3.1343
3	i	-1486.3619	6.1612	-1543.7089	3.1458	0.0049	3.1432
	f	-1486.3560	6.1618	-1543.7043	3.1514	0.0050	3.1333
4	i	-1476.8279	-143.1977	-1545.3324	3.1444	0.0019	3.3169
	f	-1476.8232	-143.1928	-1545.3286	3.1523	0.0022	3.3063
5	i	-1556.0737	-144.4279	-1362.9630	3.1635	0.1090	3.3199
	f	-1556.0759	-144.4235	-1362.9641	3.1719	0.1123	3.3067
6	i	-1676.2068	-51.3402	-1004.4455	3.2108	0.3488	3.0109
	f	-1676.2764	-51.3450	-1004.4588	3.2054	0.3633	3.0237
7	i	-635.2981	1512.8747	-1149.6245	2.1573	-0.1881	2.7540
	f	-635.3178	1512.9040	-1149.6347	2.1508	-0.1824	2.7492

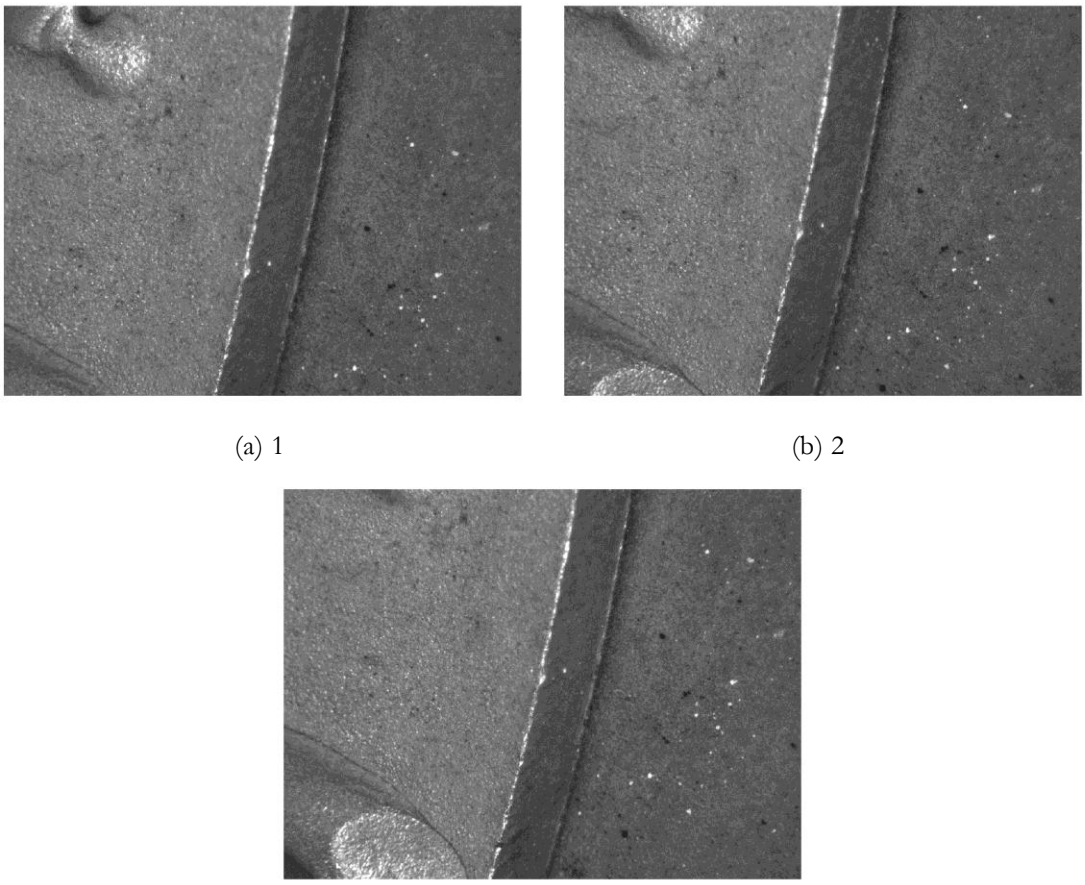
**i: initial value; f: final value.*

instances when the images were taken. Table 6.4 shows the values that were calculated from the robot feedback using the fixed coordinate transformation obtained from the calibration programs, along with the values to which the bundle adjustment algorithm converged.

6.4 Seam Tracking

After calibrating the system and testing its individual software components as described in the previous sections, performance of the integrated system on

tracking the circular flange seam was tested. Figure 6.10 shows three raw images from the image steam taken in the process.



(a) 1 (b) 2

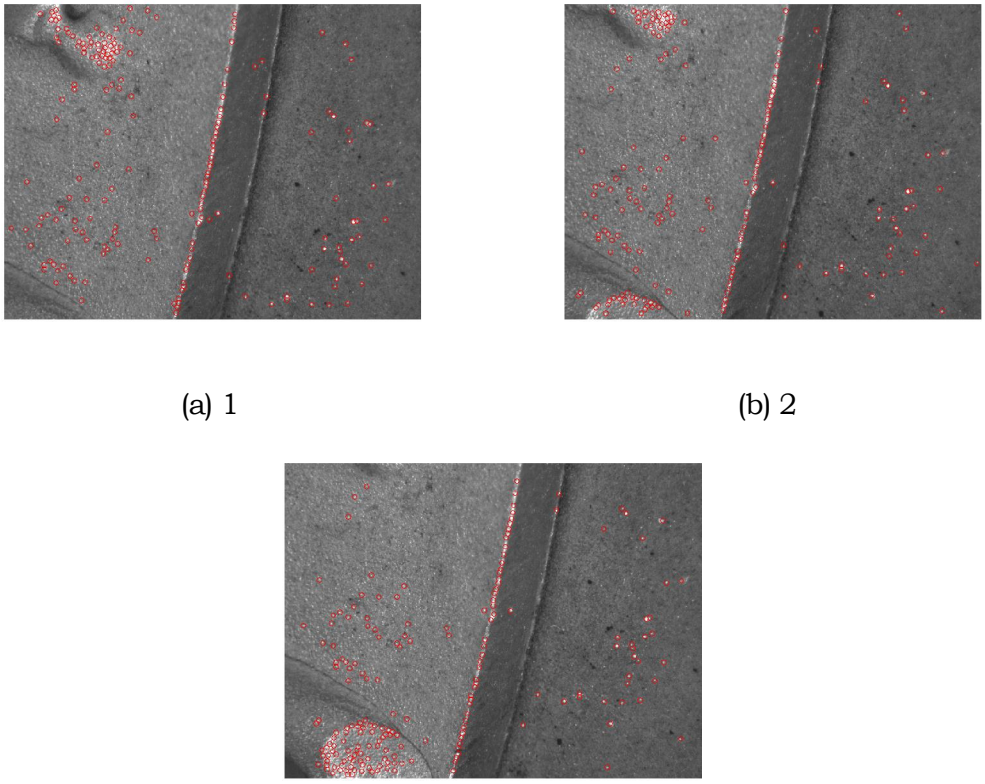
(c) 3

Figure 6.10: Flange Seam Learning Images

Figure 6.11 shows the output of the first stage in the seam tracking software: seam and feature detection using Hessian Eigenvalues.

Detected features were then filtered such that the ones far from the seam were excluded as shown in figure 6.12.

As shown in figure 6.13, a feature tracking step followed. It involved searching along the epipolar line of the images for features that minimize the region intensity difference as described in section 5.4 (page 77).



(a) 1 (b) 2

(c) 3

Figure 6.11: Seam and Feature Detection

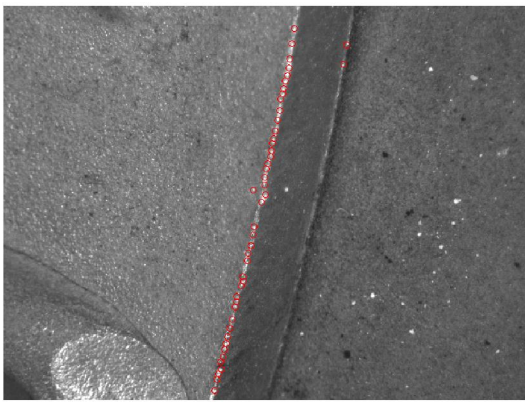


Figure 6.12: Detected Features after Excluding Non-Seam Keypoints

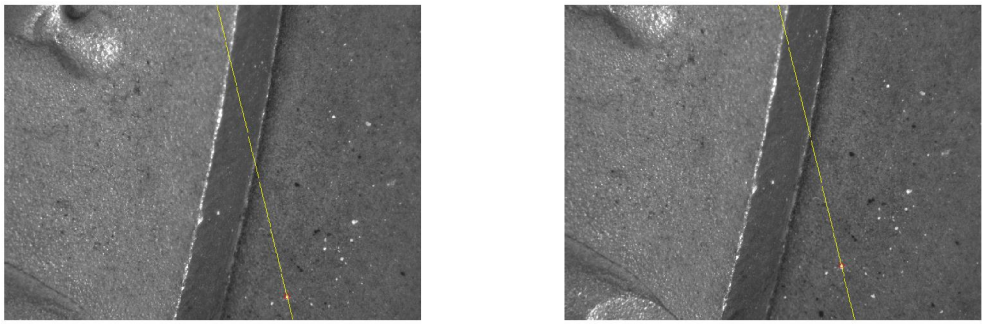


Figure 6.13: Tracking Features by Searching Along Epipolar Lines

Finally, triangulation and bundle adjustment processes followed to obtain the world positions of the seam feature points and, consequently, the three-dimensional geometry of the seam. The result is shown in figure 6.14.

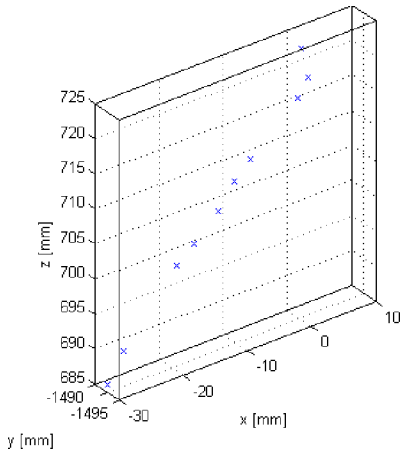


Figure 6.14: Reconstructed 3D Seam Feature Point

Locations This concludes the experimental results chapter.

Chapter 7

Discussion

The main aim of this work was to develop a direct vision-based seam tracking system. The requirements that the system needed to meet in order to perform vision-based seam tracking were identified to be acquiring images of the seam area, detecting the seam, mapping the seam in the two-dimensional images into the three-dimensional seam, and generating the necessary robot paths. The system that has been developed and tested has met these requirements through preprocessing, seam and feature detection and reconstruction by bundle adjustment.

The first part of the hypothesis on which the research was based is that seams exhibit rapid changes in optical characteristics that can be used for detecting their positions in 2D images. This has been confirmed, and the seam could be detected by various methods. Of these, the Hessian Eigenvalue method of detecting the seam was adopted in the current work because initial tests proved it more robust than filtering with difference kernels and using Canny's edge detector. Additional improvements can be saught in the future by investigating combining this method with more heuristics and implementing a contour following program. Also, a model-based approach can be followed in the final stages of

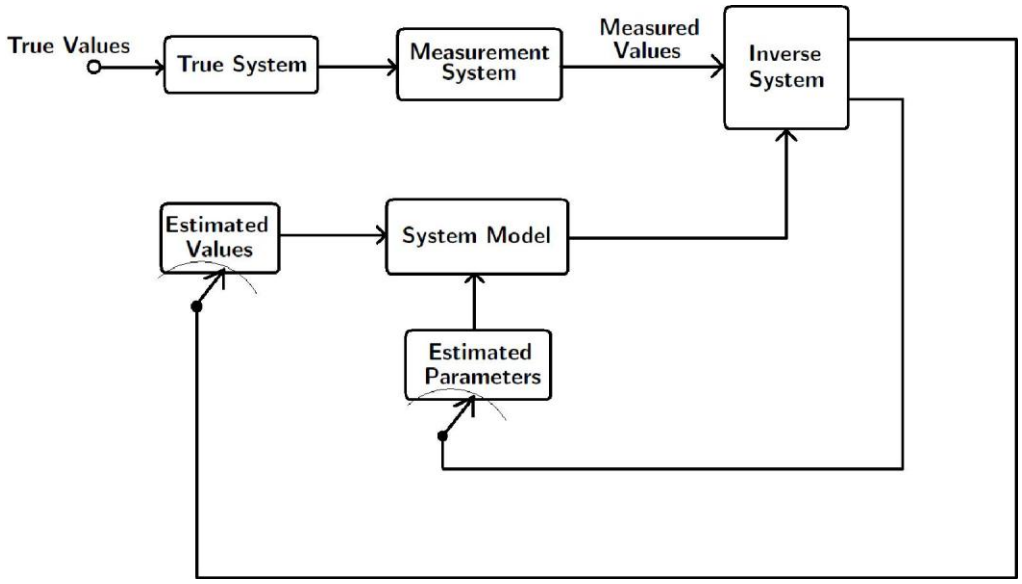


Figure 7.1: Generalized Architecture Underlying Reprojection Error Minimization

seam learning in welding applications where design of the workpieces is known beforehand but their placement for welding is not accurately controlled.

The second part of the hypothesis which involves obtaining three-dimensional information from two-dimensional images was also confirmed. After calibrating the camera, triangulation followed by bundle adjustment successfully and accurately determined the position of the detected image features.

Overall, the results described in the previous chapter confirm direct vision as a suitable strategy for optical seam teaching and tracking. The method followed for calibration and reconstruction of scene points serves as an extensible framework that is useful in many applications in addition to robotic vision. Its underlying architecture, which is described in figure 7.1, offers accurate estimation of measured values as well as model parameters. In the current application, the measured values were three-dimensional scene point positions and the model

parameters were the camera internal and external parameters for the multiple images. As an inverse model in this architecture, nonlinear least squares which was used in the current system is effective and easily adaptable to a large variety of applications.

In addition to meeting the seam reconstruction requirement, the bundle adjustment process tested in the current project gave rise to highly accurate corrections to robot position feedback. This makes robot calibration a problem that can be addressed in future projects using the current system. The estimation architecture described above could serve as a starting point for modifying the system for robot calibration. Robot joint position feedback values could be added to the values to be estimated. Additional sensed values, such as joint motor currents to indicate acceleration, could also be included. Further work is needed, however, to incorporate dynamics to the system model and solve for its inverse.

7.1 Future Research

For long welds, seam teaching alone is not sufficient, and online seam tracking is required to cope with deformations due to the welding heat input and with robot tracking errors. Further work is needed to test the possibility of accelerating the techniques investigated here for online seam tracking. Feature tracking, which required several minutes of processing per image in the reported system, readily lends itself to acceleration with a graphical processing unit (GPU) as well as implementation with a more efficient data structure. Furthermore, expanding the models used with including system dynamics could both reduce processing time as well as increase accuracy, since initial values of searches and iterative optimization modules will be nearer to their solutions.

Despite the potential of using direct vision as a strategy for optical seam tracking, many issues need to be addressed before a system can be implemented in a production plant. Specular reflections and mirror finish surfaces remain a challenge for such vision systems. Also, optical interference from the weld torch needs to be overcome. A possible method for that is capturing images at instances with low torch light intensity during the weld cycle, as demonstrated by Bae, Lee and Ahn [13]. It is also possible to use optical filters according to the welding light frequency spectrum. For MIG welding, the spectrum characterization reported by Agapiou, Kasiouras and Serafetinides [14] can be used.

The estimation architecture described in figure 7.1 enables simultaniously using multiple sensors. Using a similar architecture, it is possible to investigate the use of multiple sensor families together for seam teaching and tracking to improve performance.

Bibliography

[1] Masubuchi K. (1980). Analysis of Welded Structures. Pergamon Press; Oxford.

[2] Fu KS, Gonzalez RC, Lee CSG. (1987). Robotics: Control, Sensing, Vision and Intelligence. McGraw-Hill Book Company; New York.

[3] Bae, K.Y, Park, J.H. (2006). A study on development of inductive sensor for automatic weld seam tracking. Journal of Materials Processing Technology, 176 (1), 111-116.

[4] Jieyu, J.W, Mohanamurthy, P.H, Mun Kee, L.F, Devanthan, R, Xiaoqi, C, Piu, C.S. (2000). Development of a Closed-loop Through-the-Arc-Sensing Controller for Seam Tracking in Gas Metal Arc Welding. SIMTech Technical Report. AT/00/013/AMP.

[5] Mahajan, A, Figueroa, F. (1997). Intelligent Seam Tracking Using Ultrasonic Sensors for Robotic Welding. Robotica, 15 (1), 275-281. Cambridge University Press.

[6] Razban, A, Davies, B.L, Harris, S, Efstathiou, J. (1995). Control of an automated dispensing cell with vision controlled feedback. Control Engineering

Practice, 3 (9), 1217 - 1223.

[7] Kim, J. S, Son, Y.T, Cho, H.S, Koh, K.I. (1996). A robust visual seam tracking system for robotic arc welding. Mechatronics, 6 (2), 141 - 163.

[8] Yu, J.Y, Na, S.J. (1997). A Study on Vision Sensors for Seam Tracking of Height-Varying Weldment. Part 1: Mathematical Model. Mechatronics, 7 (7), 599 - 612.

[9] Kim, P, Rhee, S, Lee, C. H. (1999). Automatic teaching of welding robot for free-formed seam using laser vision sensor. Optics and Lasers in Engineering. 31 (1), 173 - 182.

[10] Lee, S.K, Chang, W.S, Yoo, W.S, Na, S.J. (2000). A Study on a Vision Sensor Based Laser Welding System for Bellows. Journal of Manufacturing Systems, 19 (4), 249 - 255.

[11] Yan, Z, Xu, D. (2008). Visual Tracking System for the Welding of Narrow Butt Seams in Container Manufacturing. Proceedings of the UKACC International Conference on Control. ISBN 978-0-9556152-1-4.

[12] Xu, P, Tang, X, Yao, S. (2008). Application of circular laser vision sensor (CLVS) on welded seam tracking. Journal of Materials Processing Technology, 205 (1), 404 - 410.

[13] Bae, K.Y, Lee, T.H, Ahn, K.C. (2002) An optical sensing system for seam tracking and weld pool control in gas metal arc welding of steel pipe. Journal of Materials Processing Technology, 120 (1), 458 - 465.

[14] Agapiou, G, Kasiouras, C, Serafetinides, A.A. (1999) A detailed analysis of the MIG spectrum for the development of laser-based seam tracking sensors. Optics and Laser Technology, 31 (1), 157 - 161.

[15] Murugan, N, Parmar, R.S. (1994). Effects of MIG process parameters on the geometry of the bead in the automatic surfacing of stainless steel. Journal of Materials Processing Technology, 41 (1), 381-398.

[16] Randhawa, H.S, Ghosh, P.K, Gupta, S.R. (2000). Some Basic Aspects of Geometrical Characteristics of Pulsed Current Vertical-up GMA Weld. Iron and Steel Institute of Japan (ISIJ) International, 40 (1), 71-76.

[17] Hu, J, Tsai, H.L. (2007). Heat and mass transfer in gas metal arc welding. Part II: The metal. International Journal of Heat and Mass Transfer, 50 (1), 808-820.

[18] Hu, J, Tsai, H.L. (2007). Heat and mass transfer in gas metal arc welding. Part I: The arc. International Journal of Heat and Mass Transfer, 50 (1), 833-846.

[19] Goyal, V.K, Ghosh, P.K, Saini, J.S. (2009). Analytical studies on thermal behaviour and geometry of weld pool in pulsed current gas metal arc welding. Journal of Materials Processing Technology, 209 (1), 1318-1336.

[20] Modenesi, P.J., Reis, R.I. (2007). A model for melting rate phenomena in GMA welding. Journal of Materials Processing Technology, 189 (1), 199-205.

[21] Karadeniz, E, Ozsarac, U, Yildiz, C. (2007). The effect of process parameters on penetration in gas metal arc welding processes. Materials and Design, 28 (1), 649-656.

[22] Ghosh, P.K, Dorn, L, Hubner, M, Goyal, V.K. (2007). Arc characteristics and behaviour of metal transfer in pulsed current GMA welding of aluminium alloy. Journal of Materials Processing Technology, 194 (1), 163-175.

[23] Kumar, A, Sundarrajan, S. (2009). Optimization of pulsed TIG welding process parameters on mechanical properties of AA 5456 Aluminium alloy weldments. Materials and Design, 30 (1), 1288-1297.

[24] Moon, H.S, Na, S.J. (1996). A Neuro-Fuzzy Approach to Select Welding Conditions for Welding Quality Improvement in Horizontal Fillet Welding. Journal of Manufacturing Systems, 15 (6), 392 - 403.

[25] Lee, J, Um, K. (2000). A comparison in a back-bead prediction of gas metal arc welding using multiple regressing analysis and artificial neural network. Optics and Lasers in Engineering. 34 (1), 149 - 158.

[26] Kim, I.S, Son, J.S, Lee, S.H, Yarlagadda, P.K.D.V. (2004). Optimal design of neural networks for control in robotic arc welding. Robotics and Computer-Integrated Manufacturing, 20 (1), 57-63.

[27] Kim, I.S, Son, J.S, Park, C.E, Kim, I.J, Kim, H.H. (2005). An investigation into an intelligent system for predicting bead geometry in GMA welding process. Journal of Materials Processing Technology, 159 (1), 113 - 118.

[28] Lee, C.H, Chang, K.H, Jang, G.C, Lee, C.Y. (2008). Effect of weld geometry on the fatigue life of non-load-carrying fillet welded cruciform joints. Engineering Failure Analysis. doi: 10.1016.

[29] Kolhe, K.P, Datta, C.K. (2008). Prediction of microstructure and mechanical properties of multipass SAW. Journal of Materials Processing Technology,

197 (1), 241 - 249.

[30] Deng, D, Murakawa, H. (2008). Prediction of welding distortion and residual stress in a thin plate butt-welded joint. Computational Materials Science, 43 (1), 353 - 365.

[31] Zhang, H.J, Zhang, G.J, Cai, C.B, Gao, H.M, Wu, L. (2009). Numerical simulation of three-dimension stress field in double-sided double arc multi-pass welding process. Materials Science and Engineering A, 499 (1), 309 - 314.

[32] Wang, G, Warren Liao, T. (2002). Automatic identification of different types of welding defects in radiographic images. NDT,E International, 35 (1), 519 - 528.

[33] Mirapeix, J, Garcia-Allende, P.B, Cobo, A, Conde, O.M, Lopez-Higuera, J.M. (2007). Real-time arc-welding defect detection and classification with principal component analysis and artificial neural networks. NDT,E International, 40 (1), 315 - 323.

[34] Zhiyong, L, Bao, W, Jingbin, D. (2009). Detection of GTA welding quality in disturbance factors with spectral signal of arc light. Journal of Materials Processing Technology, doi: 10.1016

[35] Dedicated Arc Welding Robot [Webpage on the Internet]. ABB Robotics. 2008. [cited 2009 June 12]. Available from: http://www.abb.com/product/seitp327/dc4d20315f67edd5c12572d500337675.aspx

[36] KUKA Robot Group. (2008). KR5 Arc HW: Specification. KUKA Roboter GmbH. Augsburg, Germany.

[37] Yaskawa Company. (2009). Motoman MA1400: Arc Welding. Yaskawa Company. Kitakyushu, Japan.

[38] Norberto Pirex, F. Godinho, T., Ferreira, P. (2004). CAD interface for automatic robot welding programming. Industrial Robot: An International Journal, 31 (1), 71 - 76.

[39] Laudridsen, J.K, Madsen, O, Holm, H. Hafsteinsson, I, Boelskifte, J. (1996). Model based control of a one degree of freedom workpiece manipulator for welding of nozzles. Mathematics and Computers in Simulation, 41 (1), 407-417.

[40] Lanzetta, M, Santochi, M, Tantussi, G. (2001). On-line control of robotized Gas Metal Arc Welding. CIRP Annals - Manufacturing Technology, 50 (1), 13 - 16.

[41] Smartt, H.B, Kenney, K.L., Charles, R.Tolle. (2002). Intelligent Control of Modular Robotic Welding Cell, Procedings of the 6th International Conference on Trends in Welding Research. ASM International, ISBN-13: 978-0871707802

[42] Bauchspiess, A, Absi Alfaro, S.C, Dobrzanski, L.A. (2001). Predictive sensor guided robotic manipulators in automated welding cells. Journal of Materials Processing Technology, 109 (1), 13 - 19.

[43] Steele, J.P.H, Mnich, C, Debrunner, C, Vincent, T, Liu, S. (2005). Development of closed-loop control of robotic welding processes. Industrial Robot: An International Journal, 32(4), 350-355.

[44] Ngo, M.D, Duy, V.H, Phuong, N.T, Kim, H.K, Kim, S.B. (2007). Develop-

ment of digital gas metal arc welding system. Journal of Materials Processing Technology, 189 (1), 384-391.

[45] Hartley R, Zisserman A. Multiple View Geometry in Computer Vision. Cambridge University Press; Cambridge: 2003

[46] Bolmsjo, G, Olsson M, Cederberg P. (2002) Robotic Arc Welding: Trends and Developments for Higher Autonomy. Industrial Robot: An International Journal, 29 (2), 98 - 104.

[47] Sonka M, Hlavac V, Boyle R. (1999). Image Processing, Analysis and Machine Vision. 2nd ed. Brooks/Cole Publishing; Pacific Grove, CA.

[48] Petrou M, Bosdogianni P. (1999). Image Processing: The Fundamentals. John Wiley, Sons; New York.

[49] Torras C. (1999). Computer Vision: Theory and Industrial Application. Springer-Verlag; New York.

[50] Gonzalez RC, Woods, RE, Eddins SL. (2004). Digital Image Processing Using Matlab. Prentice Hall; Upper Saddle River, NJ.

[51] Howard A, Rorres C. (2005). Elementary Linear Algebra. Wiley; Hoboken, NJ.

[52] Canny JF. (1983). Finding Edges and Lines in Images. MIT Press; Cambridge, MA.

[53] Szeliski R. (2010). Computer Vision: Algorithms and Applications. Springer-Verlag; New York. [Available online from http://szeliski.org/book]

[54] KUKA Robot Group [Compact Disk]. (2005). Ethernet KRL Software Module. KUKA Roboter GmbH. Augsburg, Germany.

[55] Triggs B, McLauchlan P, Hartley R, Fitzgibbon A. (2000). Bundle Adjustment – A Modern Synthesis. In Zisserman A, Szeliski R (Eds.), Vision Algorithms: Theory & Practice. Springer-Verlag; New York: 2000

[56] Harris C, Stephens M. (1988). A Combined Corner and Edge Detector. In Alvey Vision Conference. Pages 147–145.

Appendix A

Code

A.1 Ray Intersection Jacobian and Error Vector Calculation

```
function [dsdf dsdKc dsdx dsdy dsdz dsda dsdb dsdc dsdx0 dsdy0 s] = ...
    Derivatives(parameters, c1Tc2, pim1, pim2)
%This procedure accepts the image point values for a set of two images and
%a set of initial parameters for the camera and robot callibration and
%returns a vector whose components are the derivative of the distance
%between the planes containing the image rays with respect to the
%derivative parameters.

f = parameters(1);
Kc = parameters(2);
X = parameters(3); Y = parameters(4); Z = parameters(5);
A = parameters(6); B = parameters(7); C = parameters(8);
px0 = parameters(9); py0 = parameters(10);
```

```
xi1 = pim1(1); yi1 = pim1(2);
xi2 = pim2(1); yi2 = pim2(2);
%variables for sines and cosines
ca = cos(A); sa = sin(A);
cb = cos(B); sb = sin(B);
cc = cos(C); sc = sin(C);

%Inverse of an orthonormal matrix is the transpose(RPY transformation)
T =    [    cb*ca            cb*sa          -sb      0;...
          sc*sb*ca-cc*sa   sc*sb*sa+cc*ca   sc*cb    0;...
          cc*sb*ca+sc*sa   cc*sb*sa-sc*ca   cc*cb    0;...
             0                0                0      1]...
             *[1 0 0 -X;  0 1 0 -Y; 0 0 1 -Z;  0 0  0 1];

%Derivatives of the position correction matrix
dTda =  [   -cb*sa            cb*ca            0      0;...
       -sc*sb*sa-cc*ca   sc*sb*ca-cc*sa      0      0;...
       -cc*sb*sa+sc*ca   cc*sb*ca+sc*sa      0      0;...
            0                 0                0      0]...
            *[1 0 0 -X;  0 1 0 -Y; 0 0 1 -Z;  0 0  0 1];


dTdb =  [   -sb*ca           -sb*sa           -cb      0;...
          sc*cb*ca         sc*cb*sa        -sc*sb     0;...
          cc*cb*ca         cc*cb*sa        -cc*sb     0;...
             0                 0                0      0]...
             *[1 0 0 -X;  0 1 0 -Y; 0 0 1 -Z;  0 0  0 1];
```

```
dTdc = [     0                  0                0       0;...
        cc*sb*ca+sc*sa     cc*sb*sa-sc*ca     cc*cb     0;...
       -sc*sb*ca+cc*sa    -sc*sb*sa-cc*ca    -sc*cb     0;...
                 0                  0                0       0]...
                 *[1 0  0 -X; 0 1 0 -Y;  0 0  1 -Z; 0 0  0 1];


dTdx = [         0                  0                0   -cb*ca          ;...
                 0                  0                0    -sc*sb*ca+cc*sa ;...
                 0                  0                0    -cc*sb*ca-sc*sa ;...
                 0                  0                0          0         ];


dTdy = [         0                  0                0   -cb*sa          ;...
                 0                  0                0    -sc*sb*sa-cc*ca ;...
                 0                  0                0    -cc*sb*sa+sc*ca ;...
                 0                  0                0          0         ];


dTdz = [         0                  0                0          sb   ;...
                 0                  0                0   -sc*cb      ;...
                 0                  0                0   -cc*cb      ;...
                 0                  0                0          0   ];

T1_2 = T*c1Tc2*inv(T);

d1T2da  =  dTda*c1Tc2*inv(T)-T*c1Tc2*inv(T)*dTda*inv(T);
d1T2db  =  dTdb*c1Tc2*inv(T)-T*c1Tc2*inv(T)*dTdb*inv(T);
d1T2dc = dTdc*c1Tc2*inv(T)-T*c1Tc2*inv(T)*dTdc*inv(T);
```

```
d1T2dx = dTdx*c1Tc2*inv(T)-T*c1Tc2*inv(T)*dTdx*inv(T);
d1T2dy = dTdy*c1Tc2*inv(T)-T*c1Tc2*inv(T)*dTdy*inv(T);
d1T2dz = dTdz*c1Tc2*inv(T)-T*c1Tc2*inv(T)*dTdz*inv(T);

%the vectors connecting the focal points to the scene image points
r1 = [0 0 f 1]' - [1/Kc*(xi1-px0) 1/Kc*(yi1-py0) 0 1]';
r2 = T1_2 * ([0 0 f 1]' - [1/Kc*(xi2-px0) 1/Kc*(yi2-py0) 0 1]');
r1(4) = []; r2(4) = [];
%Cross product requires vectors with 3 components

%derivatives of the ray line vectors with respect to the applicable
%parameters
dr1df = [0 0 1 0]';
dr1dx0 = [1/Kc 0 0 0]';
dr1dy0 = [0 1/Kc 0 0]';
dr1dKc = (1/Kc)"2*[xi1-px0 yi1-py0 0 0]';

dr2df = T1_2*[0 0 1 0]';
dr2dx0 = T1_2*[1/Kc 0 0 0]';
dr2dy0 = T1_2*[0 1/Kc 0 0]';
dr2dKc = T1_2*((1/Kc)"2*[xi2-px0 yi2-py0 0 0]');

dr2da = d1T2da * ([0 0 f 1]' -[1/Kc*(xi2-px0) 1/Kc*(yi2-py0) 0 1]');
dr2db = d1T2db * ([0 0 f 1]' -[1/Kc*(xi2-px0) 1/Kc*(yi2-py0) 0 1]');
dr2dc = d1T2dc * ([0 0 f 1]' -[1/Kc*(xi2-px0) 1/Kc*(yi2-py0) 0 1]');

%Derivatives of r1 r2
dr1cr2df = cross(dr1df(1:3),r2(1:3))+cross(r1(1:3),dr2df(1:3));
dr1cr2dKc = cross(dr1dKc(1:3),r2(1:3))+cross(r1(1:3),dr2dKc(1:3));
dr1cr2da = cross(r1(1:3),dr2da(1:3));
```

```
dr1cr2db = cross(r1(1:3),dr2db(1:3));
dr1cr2dc = cross(r1(1:3),dr2dc(1:3));
dr1cr2dx0 = cross(dr1dx0(1:3),r2)+cross(r1(1:3),dr2dx0(1:3));
dr1cr2dy0 = cross(dr1dy0(1:3),r2)+cross(r1(1:3),dr2dy0(1:3));

%Derivatives of ||r1 r2||
dLr1cr2df = dot(cross(r1,r2),dr1cr2df)/norm(cross(r1,r2));
dLr1cr2dKc = dot(cross(r1,r2),dr1cr2dKc)/norm(cross(r1,r2));
dLr1cr2da = dot(cross(r1,r2),dr1cr2da)/norm(cross(r1,r2));
dLr1cr2db = dot(cross(r1,r2),dr1cr2db)/norm(cross(r1,r2));
dLr1cr2dc = dot(cross(r1,r2),dr1cr2dc)/norm(cross(r1,r2));
dLr1cr2dx0 = dot(cross(r1,r2),dr1cr2dx0)/norm(cross(r1,r2));
dLr1cr2dy0 = dot(cross(r1,r2),dr1cr2dy0)/norm(cross(r1,r2));

%the vector normal to both the ray vectors and its derivative
nh = cross(r1,r2); nh = nh/norm(nh);

dnhdf = (dr1cr2df*norm(cross(r1,r2))-cross(r1,r2)*dLr1cr2df)...
    /(norm(cross(r1,r2)))"2;
dnhdKc = (dr1cr2dKc*norm(cross(r1,r2))-cross(r1,r2)*dLr1cr2dKc)...
    /(norm(cross(r1,r2)))"2;
dnhda = (dr1cr2da*norm(cross(r1,r2))-cross(r1,r2)*dLr1cr2da)...
    /(norm(cross(r1,r2)))"2;
dnhdb = (dr1cr2db*norm(cross(r1,r2))-cross(r1,r2)*dLr1cr2db)...
    /(norm(cross(r1,r2)))"2;
dnhdc = (dr1cr2dc*norm(cross(r1,r2))-cross(r1,r2)*dLr1cr2dc)...
    /(norm(cross(r1,r2)))"2;
dnhdx0 = (dr1cr2dx0*norm(cross(r1,r2))-cross(r1,r2)*dLr1cr2dx0)...
    /(norm(cross(r1,r2)))"2;
dnhdy0 = (dr1cr2dy0*norm(cross(r1,r2))-cross(r1,r2)*dLr1cr2dy0)...
```

```
    /(norm(cross(r1,r2)))^2;


%The vector connecting the two rays which is perpendicular to both
s = nh*dot([nh; 0],([0 0 f 1]'-T1-2*[0 0 f 1]'));
Ls = abs(dot([nh; 0],([0 0 f 1]'-T1-2*[0 0 f 1]')));

dsdf = dnhdf*(dot([nh; 0],([0 0 f 1]'-T1-2*[0 0 f 1]')))+nh*...
    (dot([dnhdf; 0],[0 0 f 1]'-T1-2*[0 0 f 1]')+dot([nh; 0],...
    [0 0 1 0]'-T1-2*[0 0 1 0]'));
dsdKc = dnhdKc*(dot([nh; 0],([0 0 f 1]'-T1-2*[0 0 f 1]')))+nh*...
    (dot([dnhdKc; 0],[0 0 f 1]'-T1-2*[0 0 f 1]'));
dsda = dnhda*(dot([nh; 0],([0 0 f 1]'-T1-2*[0 0 f 1]')))+nh*...
    (dot([dnhda; 0],[0 0 f 1]'-T1-2*[0 0 f 1]')-dot([nh; 0],...
    d1T2da*[0 0 f 1]'));
dsdb = dnhdb*(dot([nh; 0],([0 0 f 1]'-T1-2*[0 0 f 1]')))+nh*...
    (dot([dnhdb; 0],[0 0 f 1]'-T1-2*[0 0 f 1]')-dot([nh; 0],...
    d1T2db*[0 0 f 1]'));
dsdc = dnhdc*(dot([nh; 0],([0 0 f 1]'-T1-2*[0 0 f 1]')))+nh*...
    (dot([dnhdc; 0],[0 0 f 1]'-T1-2*[0 0 f 1]')-dot([nh; 0],...
    d1T2dc*[0 0 f 1]'));
dsdx = nh*(-dot([nh; 0],d1T2dx*[0 0 f 1]')); dsdy =
nh*(-dot([nh; 0],d1T2dy*[0 0 f 1]')); dsdz = nh*(-
dot([nh; 0],d1T2dz*[0 0 f 1]')); dsdx0 =
dnhdx0*(dot([nh; 0],([0 0 f 1]'-T1-2*[0 0 f 1]')))+...
        nh*(dot([dnhdx0; 0],[0 0 f 1]'-T1-2*[0 0 f 1]'));
dsdy0 = dnhdy0*(dot([nh; 0],([0 0 f 1]'-T1-2*[0 0 f 1]')))+...
        nh*(dot([dnhdy0; 0],[0 0 f 1]'-T1-2*[0 0 f 1]'));

dLsdf = dot(s,dsdf)/norm(s);
```

```
dLsdKc = dot(s,dsdKc)/norm(s);
dLsda = dot(s,dsda)/norm(s);
dLsdb = dot(s,dsdb)/norm(s);
dLsdc = dot(s,dsdc)/norm(s);
dLsdx = dot(s,dsdx)/norm(s);
dLsdy = dot(s,dsdy)/norm(s);
dLsdz = dot(s,dsdz)/norm(s);
dLsdx0 = dot(s,dsdx0)/norm(s);
dLsdy0 = dot(s,dsdy0)/norm(s);
```

A.1.1 Ray Intersection Error: Single Iteration

```
% File: RayErrorAdjustmentIteration.m
% Performs one gradient descent iteration
% Takes the Jacobian matrix and error vector and generates
% the corresponding adjustment vector (xh)

A = [];
S = [];

camTgripper = eye(4);

for pt = 1:size(correspondence,1) % for each point
    for im1 = 1:(size(correspondence,2)-1) % for each image
        for im2 = im1+1:size(correspondence,2) % for each other image
            [dLsdf dLsdKc dLsdx dLsdy dLsdz dLsda...
                dLsdb dLsdc dLsdx0 dLsdy0 Ls]...
                = Derivatives(InitialValues, camTgripper...
```

```
                *camTworld(positions(im1,2:end))...
                *inv(camTworld(positions(im2,2:end)))...
                *inv(camTgripper)...
                , reshape(correspondence(pt,im1,[2 3]),1,2)...
                , reshape(correspondence(pt,im2,[2 3]),1,2));
            A = [A; dLsdf... dLsdKc
                dLsdx dLsdy dLsdz dLsda dLsdb dLsdc dLsdx0 dLsdy0];
            S = [S; Ls];
        end
    end
end

        sse = sum(S.^2)
 [u s v] = svd(A);
 sd = diag(s);
 sd(sd < 1e-4*sd(1)*ones(size(sd))) = [];
 xh = v(:,1:length(sd))*diag(1./sd)*u(:,1:length(sd))'*S;
```

A.1.2 Ray Intersection Error Adjustment Main Loop

```
% Initialize arrays
SSE = [];
IV = [];

% Initial values
f = 69.9; Kc = 285;
X = 0; Y = 0; Z = -87.36;
A = 0; B = 0; C = 0;
```

```
px0 = 517; py0 = 389;
InitialValues = [f Kc X Y Z A B C px0 py0]'

positions = csvread('C:\MEng\Calibration\Nov22Dots\positions.txt');
points = csvread('C:\MEng\Calibration\Nov22Dots\points.csv');

correspondence = zeros(max(points(:,2)),max(points(:,1)),3);
for i = 1:size(points,1)
% show that there's a point there
    correspondence(points(i,2),points(i,1),1) = 1;
    correspondence(points(i,2),points(i,1), [2 3]) = points(i,
[3 4]); end

% Gradient descent
RayErrorAdjustmentIteration;
sseOld = inf;
while(sseOld - sse > 1e-4)
    SSE = [SSE; sse];
    IV = [IV; InitialValues'];
    InitialValues = InitialValues - [xh(1); 0; xh(2:end)];
    sseOld = sse;
    RayErrorAdjustmentIteration;
end

BestCandidate = IV(end,:)'
```

A.2 Camera Transformation

```
function camTworld = camTworld(coordinates)
%accepts a coordinates matrix [X Y Z A B C] in [mm, deg] and returns
a %homogeneous transformation matrix that converts vectors from world
%coordinate frame to camera coordinate frame

X = coordinates(1); Y = coordinates(2); Z = coordinates(3);
A = deg2rad(coordinates(4));
B = deg2rad(coordinates(5));
C = deg2rad(coordinates(6));

%variables for sines and cosines
ca = cos(A); sa = sin(A);
cb = cos(B); sb = sin(B);
cc = cos(C); sc = sin(C);

%Inverse of an orthonormal matrix is the transpose(RPY transformation)
  camTworld = [cb*ca cb*sa -sb 0; sc*sb*ca-cc*sa sc*sb*sa+cc*ca sc*cb 0;
  cc*sb*ca+sc*sa cc*sb*sa-sc*ca cc*cb 0; 0 0 0 1]...
    *[1 0 0 -X; 0 1 0 -Y; 0 0 1 -Z; 0 0 0 1];
```

A.3 EthernetKRL Client Program

.

.

.

```
typedef struct dw_OperationData
  {
  /* Determines the control code. */
  long sl_CntrlCode;

  /* operation code for the operation. */
  long sl_OpCode;

  /* Number of repetetions of the operation */
  long sl_Repetetion;

  /* Operation specific data/ parameter */
  float sf_Length;

  /* Operation specific position/ parameters */
  float af_Position[6];

  /* Description of operation */
  char ac_Desc[81];

  /* Refreshed data flag */
BOOL sb_IsNewData;

  }SW_OPDATA;

.
.
.

void CEthKrlServDlg::mfv_UpdateOpGui()
```

```
{
  WaitForSingleObject(xxhx_OpDataMutex,INFINITE);
  if(xxsw_RcvdOpData.sb_IsNewData)
  {
.
.
.
    memcpy(&sw_LocalOpData, &xxsw_RcvdOpData, sizeof(SW_OPDATA));
    .
    .
    .
  }
}
```

A.4 Seam and Feature Detection

```
M = im2double(M);
if(ndims(M) == 3)
    M = (M(:,:,1) + M(:,:,2) + M(:,:,3))/3;
end
ssq = 1;
GaussianDieOff = 0.005;
pw = 1:30; %possible widths
width = find(exp(-(pw.*pw)/(2*ssq))>GaussianDieOff,1,'last');
t = (-width:width);
gau = exp(-(t.*t)/(2*ssq))/(2*pi*ssq); % the gaussian 1D filter
[x,y]=meshgrid(-width:width,-width:width); dgau2D=-x.*exp(-
(x.*x+y.*y)/(2*ssq))/(pi*ssq);
```

```
gau2D=exp(-(x.*x+y.*y)/(2*ssq))/(pi*ssq);
gau2D=gau2D/sum(gau2D(:));
%aSmooth = M;
aSmooth=imfilter(M,gau,'conv','replicate');% run the filter accross rows
aSmooth=imfilter(M,gau','conv','replicate'); % run the filter accross rows
ax = imfilter(aSmooth, dgau2D, 'conv','replicate');
ay = imfilter(aSmooth, dgau2D', 'conv','replicate');
mag = sqrt((ax.*ax) + (ay.*ay));
magmax = max(mag(:));
mag = mag/magmax;

% Finding the features

axx = ax.*ax; ayy = ay.*ay; axy
= ax.*ay; imA = reshape([axx,
ayy, axy], [size(axx,1)
size(axx,2) 3]);

[x2 y2] = meshgrid(-2*width:2*width,-2*width:2*width);
gau2D = exp(-(x2.*x2 + y2.*y2)/(8*ssq))/(4*pi*ssq);
imAf = imfilter(imA,gau2D,'conv','replicate');
% Smaller eigenvalue
Lmin = 1/2*((imAf(:,:,1)+imAf(:,:,2))-((imAf(:,:,1)+imAf(:,:,2))."2 ...
            -4*(imAf(:,:,1).*imAf(:,:,2)-imAf(:,:,3))."2))."0.5);
Lmax = 1/2*((imAf(:,:,1)+imAf(:,:,2))+((imAf(:,:,1)+imAf(:,:,2))."2 ...
            -4*(imAf(:,:,1).*imAf(:,:,2)-imAf(:,:,3))."2))."0.5);

detAf = imAf(:,:,1).*imAf(:,:,2) - imAf(:,:,3)."2;
trAf = imAf(:,:,1)+imAf(:,:,2);
%q = detAf - 0.06*trAf."2;
```

```
q = Lmin;

figure, subplot(2,2,1), imshow(M), title('Original Image')
subplot(2,2,2), imshow(q/max(q(:))), title('Trackable Features')
subplot(2,2,3), imshow(Lmax/max(Lmax(:))), title('Highest Eigenvalue')
Moverlay = reshape([q/max(q(:)) zeros(size(q)) 0.25*M],size(q,1),[],3);
subplot(2,2,4), imshow(Moverlay), title('Overlay')

% finding local maxima
[xi yi] = meshgrid(1:size(q,2),1:size(q,1)); % grid values of the image
iLM = []; % indeces of the local maxima
Features = []; % Features
rN = 10; % radius of the neighbourhood
figure
LM = zeros(size(M));
ImageCentre = find(xi == floor(size(xi,2)/2) & yi == floor(size(yi,1)/2));
NormalNeighbourhoodSize = sum(sum(((xi(:)-...
            xi(ImageCentre)*ones(length(xi(:)),1))."2+(yi(:)...
            -yi(ImageCentre)*ones(length(yi(:)),1))."2)."0.5...
        < rN*ones(length(xi(:)),1)))
for i = 1:200
    [qs I] = sort(q(:),'descend');
    mN = ((xi(:)-xi(I(1))*ones(length(xi(:)),1))."2+(yi(:)...
                -yi(I(1))*ones(length(yi(:)),1))."2)."0.5...
        < rN*ones(length(xi(:)),1); % Neighbourhood membership function
    % remove the other neighbourhood corners
    q(mN) = 0;
    % exclude features that are too close to the boundaries
    if sum(mN(:)) < NormalNeighbourhoodSize
        continue;
```

```
    end
    Features = [Features M(mN)];
    LM(I(1)) = 1;
    iLM = [iLM; I(1)];
end
Moverlay = reshape([M/6+LM M/6 M/6],size(M,1),[],3);
imshow(Moverlay)
```

A.5 Bundle Adjustment

A.5.1 Reprojection Errors and Jacobian

```
function [dedx dedy dedz dedX dedY dedZ ...
    deda dedb dedc dedf dedKc dedpx0 dedpy0 e] = ...
    BundleDerivatives(parameters, pim, Psc)
%This procedure accepts the image point values for an image and a set of
%initial parameters for the camera and robot callibration and scene point
%and returns a vector whose components are the derivatives of the error
%between the actual and reprojected image points

% Camera external and internal parameters
f = parameters(1);
Kc = parameters(2);
X = parameters(3); Y = parameters(4); Z = parameters(5);
A = parameters(6); B = parameters(7); C = parameters(8);
px0 = parameters(9); py0 = parameters(10);
```

```
%variables for sines and cosines
ca = cos(A); sa = sin(A);
cb = cos(B); sb = sin(B);
cc = cos(C); sc = sin(C);

%scene points (if not already homogenious, homogenize)
P = Psc;
if(size(P,1)~=4)
   P = [P; ones(1,size(P,2))];
end

%Inverse of an orthonormal matrix is the transpose(RPY transformation)
R =    [   cb*ca            cb*sa          -sb   ;...
         sc*sb*ca-cc*sa sc*sb*sa+cc*ca     sc*cb ;...
         cc*sb*ca+sc*sa cc*sb*sa-sc*ca     cc*cb ];

%Derivatives of the position correction matrix
dRda = [ -cb*sa            cb*ca            0 ;...
       -sc*sb*sa-cc*ca sc*sb*ca-cc*sa       0 ;...
              -cc*sb*sa+sc*ca cc*sb*ca+sc*sa      0 ];


dRdb = [ -sb*ca           -sb*sa          -cb   ;...
         sc*cb*ca         sc*cb*sa       -sc*sb ;...
         cc*cb*ca         cc*cb*sa       -cc*sb ];


dRdc = [    0                 0             0   ;...
       cc*sb*ca+sc*sa cc*sb*sa-sc*ca     cc*cb ;...
```

```
            -sc*sb*ca+cc*sa  -sc*sb*sa-cc*ca -sc*cb ];

% translation vector
t = [-X; -Y; -Z];
dtdX = [-1; 0; 0];
dtdY = [0; -1; 0];
dtdZ = [0; 0; -1];

% scene point
dPdx = [1; 0; 0; 0]*ones(1,size(P,2));
dPdy = [0; 1; 0; 0]*ones(1,size(P,2));
dPdz = [0; 0; 1; 0]*ones(1,size(P,2));

% camera matrix
K = [-Kc*f 0 px0; 0 -Kc*f py0; 0 0 1];
dKdKc = [-f 0 0; 0 -f 0; 0 0 0];
dKdf = [-Kc 0 0; 0 -Kc 0; 0 0 0];
dKdpx0 = [0 0 1; 0 0 0; 0 0 0];
dKdpy0 = [0 0 0; 0 0 1; 0 0 0];

% reprojected projective point
xp = K*[R t]*P;
dxpda = K*[dRda [0;0;0]]*P;
dxpdb = K*[dRdb [0;0;0]]*P;
dxpdc = K*[dRdc [0;0;0]]*P;
dxpdX = K*[zeros(3,3) dtdX]*P;
dxpdY = K*[zeros(3,3) dtdY]*P;
dxpdZ = K*[zeros(3,3) dtdZ]*P;
dxpdx = K*[R t]*dPdx;
dxpdy = K*[R t]*dPdy;
```

```
dxpdz = K*[R t]*dPdz;
dxpdKc = dKdKc*[R t]*P;
dxpdf = dKdf*[R t]*P;
dxpdpx0 = dKdpx0*[R t]*P;
dxpdpy0 = dKdpy0*[R t]*P;

% reprojection error
e = pim - xp(1:2,:)./([1;1]*xp(3,:));
deda = -(dxpda(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpda(3,:)))...
    ./([1;1]*xp(3,:)."2);
dedb = -(dxpdb(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdb(3,:)))...
    ./([1;1]*xp(3,:)."2);
dedc = -(dxpdc(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdc(3,:)))...
    ./([1;1]*xp(3,:)."2);
dedX = -(dxpdX(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdX(3,:)))...
    ./([1;1]*xp(3,:)."2);
dedY = -(dxpdY(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdY(3,:)))...
    ./([1;1]*xp(3,:)."2);
dedZ = -(dxpdZ(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdZ(3,:)))...
    ./([1;1]*xp(3,:)."2);
dedx = -(dxpdx(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdx(3,:)))...
    ./([1;1]*xp(3,:)."2);
dedy = -(dxpdy(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdy(3,:)))...
    ./([1;1]*xp(3,:)."2);
dedz = -(dxpdz(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdz(3,:)))...
    ./([1;1]*xp(3,:)."2);
dedKc = -(dxpdKc(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdKc(3,:)))...
    ./([1;1]*xp(3,:)."2);
dedf = -(dxpdf(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdf(3,:)))...
    ./([1;1]*xp(3,:)."2);
```

```
dedpx0 = ...
     -(dxpdpx0(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdpx0(3,:)))...
     ./([1;1]*xp(3,:)."2);
dedpy0 = ...
     -(dxpdpy0(1:2,:).*([1;1]*xp(3,:))-xp(1:2,:).*([1;1]*dxpdpy0(3,:)))...
     ./([1;1]*xp(3,:)."2);
```

A.5.2 Bundle Adjustment Iteration

```
% Bundle Adjustment
% T is the camera correction matrix
% p is the position of points (3xN)
% positions is the positions of the robot end effector for the images
% correspondence is an array of the point correspondences
% (nPoints x nImages x 3), the first nPxnI array is a weight array
%                          the second is the x-components and the
%                          third is the y-components of the image points
% f, Kc, px0 and py0 are the parameters of the camera matrix
%
% One way to achieve these requirements is to run RayErrorAdjustment
%   followed by MultiImagePoints

%p = reshape(p3D(1:3,:),3,[]);

errors = [];
dedP = []; % derivatives in terms of scene point
dedT = []; % derivatives in terms of camera position and pose
dedK = []; % derivatives in terms of camera internal parameters
```

```
for imageIndex = 1:size(correspondence,2)
    parameters = [CameraInt(1:2); CameraExt(:,imageIndex); CameraInt(3:4)];
    imgPoints = reshape(correspondence(:,imageIndex,[2 3]),[],2)';
    [dedx dedy dedz dedX dedY dedZ deda dedb...
        dedc dedf dedKc dedpx0 dedpy0 e]...
        = BundleDerivatives(parameters, imgPoints, p);
    errors = [errors; e];
    dedP = [dedP; dedx; dedy; dedz];
            dedT = [dedT; dedX; dedY; dedZ; deda; dedb; dedc];
    dedK = [dedK; dedf; dedKc; dedpx0; dedpy0];
end

% re-organization
e = reshape([reshape(errors(1:2:end,:)',[],1)';...
    reshape(errors(2:2:end,:)',[],1)'],[],1); % errors into one column

JdedK = [reshape([reshape(dedK(1:8:end,:)',[],1)';...
 ... % Jacobian of internal parameters
        reshape(dedK(2:8:end,:)',[],1)'],[],1),...
        reshape([reshape(dedK(3:8:end,:)',[],1)';...
        reshape(dedK(4:8:end,:)',[],1)'],[],1),...
        reshape([reshape(dedK(5:8:end,:)',[],1)';...
        reshape(dedK(6:8:end,:)',[],1)'],[],1),...
        reshape([reshape(dedK(7:8:end,:)',[],1)';...
        reshape(dedK(8:8:end,:)',[],1)'],[],1)];
% Jacobian of scene points
JdedP = zeros(size(e,1),3*size(correspondence,1));
% Jacobian of external parameters
JdedT = zeros(size(e,1),6*size(correspondence,2));
```

```
% populating the external parameters and scene points Jacobians
for i = 1:size(correspondence,2) % for each image
   blockStart = 2*size(correspondence,1)*(i-1);
   for pIndex = 1:size(correspondence,1) % for each point
        JdedP(blockStart+2*(pIndex-1)+1:blockStart+2*pIndex,...
                    3*(pIndex-1)+1:3*pIndex) = ...
           reshape(dedP(6*(i-1)+1:6*i,pIndex),2,3);
   end
           dedT2row = reshape(dedT(12*(i-1)+1:12*i,:),2,[]);
   JdedT(2*size(correspondence,1)*(i-1)+1:2*size(correspondence,1)*i...
        , 6*(i-1)+1:6*i)...
         = reshape([dedT2row(:,1:6:end) dedT2row(:,2:6:end)...
          dedT2row(:,3:6:end) dedT2row(:,4:6:end) dedT2row(:,5:6:end)...
          dedT2row(:,6:6:end)],[],6);
end

sse = sum(e.^2);

Jacobian = [JdedT JdedP JdedK];
[u s v] = svd(Jacobian);
sd = diag(s);
sd(sd < 1e-4*sd(1)*ones(size(sd))) = [];
        xh = v(:,1:length(sd))*diag(1./sd)*u(:,1:length(sd))'*e;
```

A.6 FeatureMatching

```
% takes feature points and performs tracking (by finding indices of
```

```
% matching features)

% features must have been found and stored with indices in iLM

% prepare the grid
[x y] = meshgrid(1:size(M,2),1:size(M,1));

matches = [];

% loop over the found features
for feature = 1:15
    %form the feature area
    featureIndex = iLM(feature);
    parentGrid = [x(featureIndex)-3:1:x(featureIndex)+3; ...
      y(featureIndex)-3:1:y(featureIndex)+3];
    area = ParentImage(parentGrid(2,:),parentGrid(2,:));
    %center and normalize the feature area
    area = (area-mean(area(:)))/std(area(:));

    %scanning start
    featureGrid = parentGrid + [-15; -74]*ones(1,size(parentGrid,2));
    matchError = inf;
    %scan for the new area
    for xOffset = -3:3
       for yOffset = -7:7
                %for theta = deg2rad(-0.1:0.025:0.1)
          theta = 0;
               currentScanGrid = [cos(theta) sin(theta) xOffset; ...
               -sin(theta) cos(theta) yOffset; 0 0 1] ...
               *[featureGrid; ones(1,size(featureGrid,2))];
```

```
                [xCurrent yCurrent] ...
                = meshgrid(currentScanGrid(1,:),currentScanGrid(2,:));
                currentArea ...
                = TrackedImage(currentScanGrid(2,:), currentScanGrid(1,:));
                currentArea ...
                = (currentArea-mean(currentArea(:)))/std(currentArea(:));

                currentError = sum((currentArea(:)-area(:)).^2);
                if(currentError < matchError)
                    matchError = currentError;
                    matchIndex = currentScanGrid(:,4);
                    matchAngle = theta;
                end
                %figure, mesh(1:size(currentArea,2),1:size(currentArea,1),cur
                %title('current area')
                %figure, mesh(1:size(area,2),1:size(area,1),area);
                %title('area')
            %end
        end
    end

    matches = [matches [matchIndex([1 2]);theta;matchError]];
end
```

Appendix B

Focal Length Calibration Data

Table B.1: Distance and Reciprocal of Imaged Dimension

Distance [mm]	Major Radius [pixels]	Reciprocal of Major Radius [1/pixels]
1482.418	126.2992719	0.007917702015
1482.05	126.4498742	0.007908272001
1481.539	126.5517664	0.007901904715
1481.2	126.6107923	0.007898220856
1480.826	126.6514275	0.007895686765
1480.381	126.7207016	0.007891370453
1479.915	126.8366236	0.007884158151
1479.553	126.8604399	0.007882678008
1479.217	126.941765	0.00787762798
1478.828	127.0073603	0.007873559436
1478.474	127.0263459	0.007872382637
1478.062	127.136799	0.007865543319
1477.55	127.2445967	0.007858879871
1477.045	127.3228153	0.007854051905
1476.514	127.4122318	0.007848540016
1475.926	127.5456728	0.007840328705

Table B.1: *Continued*

Distance [mm]	Major Radius [pixels]	Reciprocal of Major Radius [1/pixels]
1475.327	127.5967633	0.007837189396
1474.811	127.7037796	0.007830621796
1474.15	127.8471143	0.007821842559
1473.602	127.9116891	0.007817893788
1473	128.0259336	0.007810917459
1472.472	128.1121641	0.007805660038
1471.828	128.2173193	0.007799258364
1471.12	128.3463743	0.007791416042
1470.351	128.4908022	0.007782658236
1469.697	128.611724	0.00777534092
1469.146	128.7105509	0.007769370834
1468.536	128.8412839	0.007761487386
1467.713	128.956854	0.007754531603
1466.917	129.1295454	0.00774416108
1466.378	129.2405647	0.007737508749
1465.42	129.4350335	0.007725883582
1464.583	129.6090338	0.00771551157
1463.931	129.732432	0.007708172774

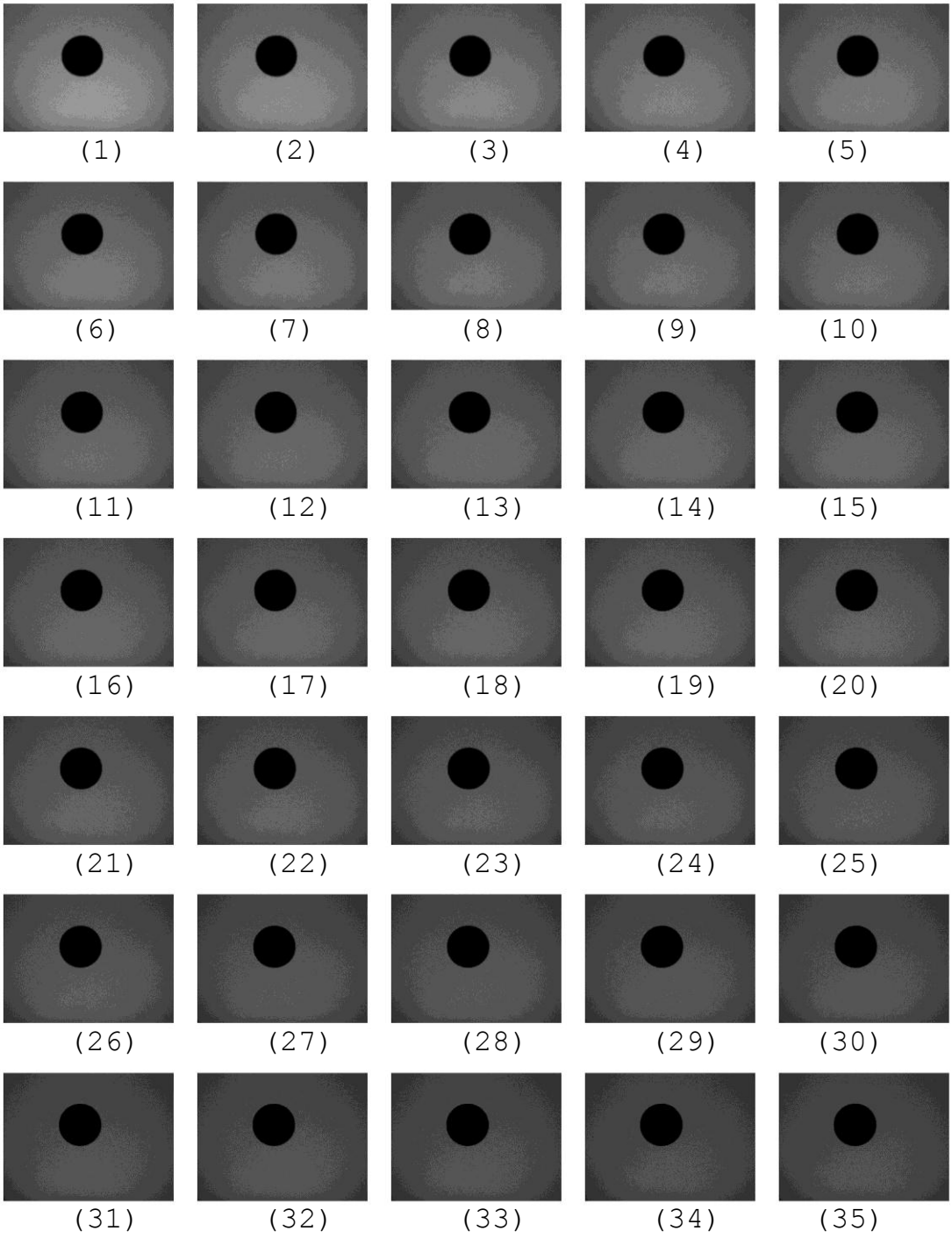


Figure B.1: Focal Length Determination Images