Dublin City University

Design and Control of a Robotic System for Application of Milking Cups

Ph.D. Thesis

Author:
Damian Christie

Supervisors:
Dr. Harry Esmonde
Dr. Brian Corcoran

November 20, 2012

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D. is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____

ID No.: 51004301

Date: November 20, 2012

Acknowledgements

I would first and foremost like to thank Harry Esmonde, my supervisor and mentor in my postgraduate studies. Harry's enthusiasm for the project and the fervor with which he pursues a good engineering problem has been an inspiration. His patience and optimism have helped maintain sight of the bigger picture when needed. His open door, from the smallest query to the most seemingly impassable problem, has ensured that I have gotten where I am.

I would also like to thank Brian Corcoran for his guidance and support on the project; from my introduction to the fairly intimidating and expensive robot I was given responsibility of, right up to my bombardment of his inbox with chapters for proof reading.

I am also indebted to the technicians of the School of Mechanical Engineering for providing support in the area of mechanical design and for their administration of resources.

I would like to thank all of the teaching staff in DCU who have contributed to my development as an engineer, as well as all of the teachers I have had through my primary and secondary education.

The most important people to be thanked are my parents, Dermot and Annette, for instilling in me the importance of education since before I started school. This has come in many forms, from teaching me to read from Ladybird books about the 'Elves and the Shoemaker' to introducing me to Star Trek, which ultimately led me to robotics. The moral, emotional and social education they have provided, and continue to provide, as well as their belief in me and my abilities have shaped who I am.

Abstract

A robotic system is developed capable of retrieving milking vacuum cups and placing them at points representing the teats of a cow. The points are identified manually and measured using a stereo vision system. Although the system is still at the laboratory development stage it is capable of manipulating the cups simultaneously and independently and of accessing the teats from between the rear legs of a cow. This sets it apart from the similar systems currently available on the market, most of which have been designed for use in shed-based dairy farming, a farming model not commonly used in larger dairy farming operations. The system has been developed to integrate a stereo vision system, in which the teats are manually identified. This is shown to provide sufficiently accurate measurements to allow the placement of the cups onto the teats. However several techniques are investigated to improve the results of an automated teat identification process (to be developed in future work) including the ability to handle the possible case where a spurious object is misidentified as a teat.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Dairy farming is a large part of the Irish economy. The value of dairy exports in 2010 was 2.3 billion [1] accounting for 29% of agricultural goods output from Ireland in 2010 [2]. It employs 20,000 farmers with an additional 4,500 employed in the processing industry [3]. There has been a growth in the average herd size in Ireland over the last 20 years to about 60 cows. This growth has been particularly concentrated on the number of farms with herds of 50 - 100 cows [4]. Dairy farming is a labour intensive process with the milking of the cows accounting for up to a third of labour costs for dairy farmers [5]. The traditional trend of migration from rural to urban areas by young people also poses a challenge to farmers when trying to find labourers to work in dairy parlours [6]. With the current economic situation adding to the pressures of increased herd size and labour costs on dairy farmers, the use of automation is becoming more appealing as a method of reducing running costs.

The automation of the milking process has been under way since the 19th century, when development began on using vacuum cups to extract the milk from the teats. Pulsating vacuum cups have been in use since the 1890s, overcoming problems of earlier systems that used either catheters, which had low yield and increased risk of infection, or used continuous vacuums, which resulted in bruising of the teats [7].

The modern milking cup has a rigid outer shell with a rubber liner inside, Figure 1.1.

The cup uses two vacuum lines, one that reduces the pressure inside the rubber liner, and one that reduces the pressure between the rubber liner and the outer shell. The vacuum inside the liner is constant, and serves to carry the milk away from the teat. The vacuum between the liner and the shell is pulsed and results in the suction inside the liner being cut-off when the vacuum is switched off [8].

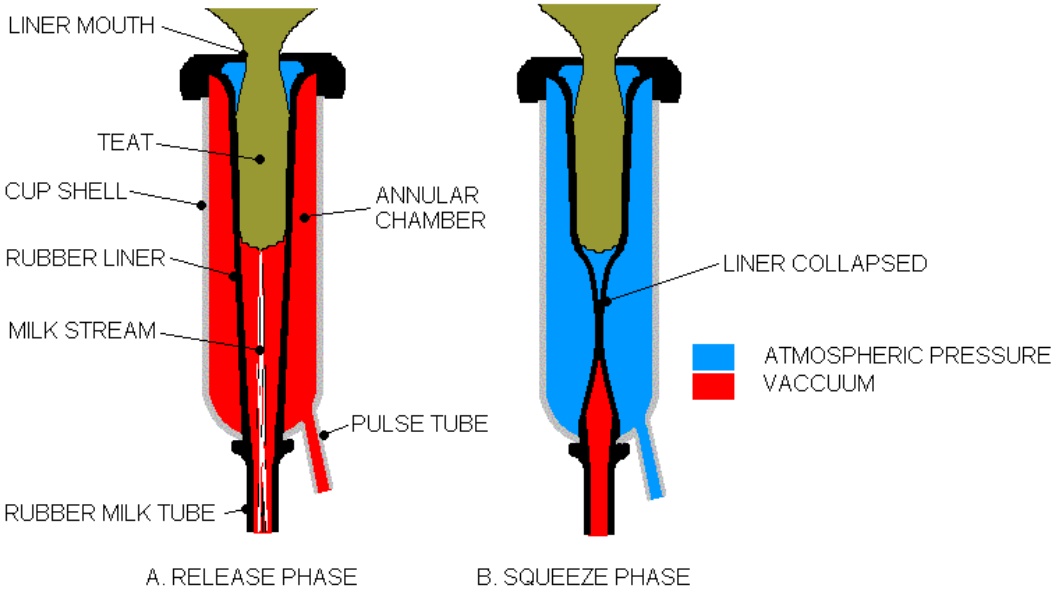


Figure 1.1: Pulsing action of milking cup [9].

Once the milk is collected, it is processed automatically in the parlour. The design of the parlour itself takes more from a modern production line than from the habitat of the animals. This enables farmers to milk large herds quickly and efficiently, yielding large quantities of milk. This system is also beneficial to the animals in that they can be returned to pasture as quickly as possible and with as little stress as possible.

1.1 Rotary Carousel Description and Manual Application Time

Two of the most common types of parlour are the herringbone and the carousel, Figure 1.2. In the herringbone parlour, the stalls are arranged side by side in straight lines.

This allows the farmer to walk between the two rows applying the cups to the cows in batches. Normal capacity for a herringbone parlour with one operator would be around 50-75 cows/hour.

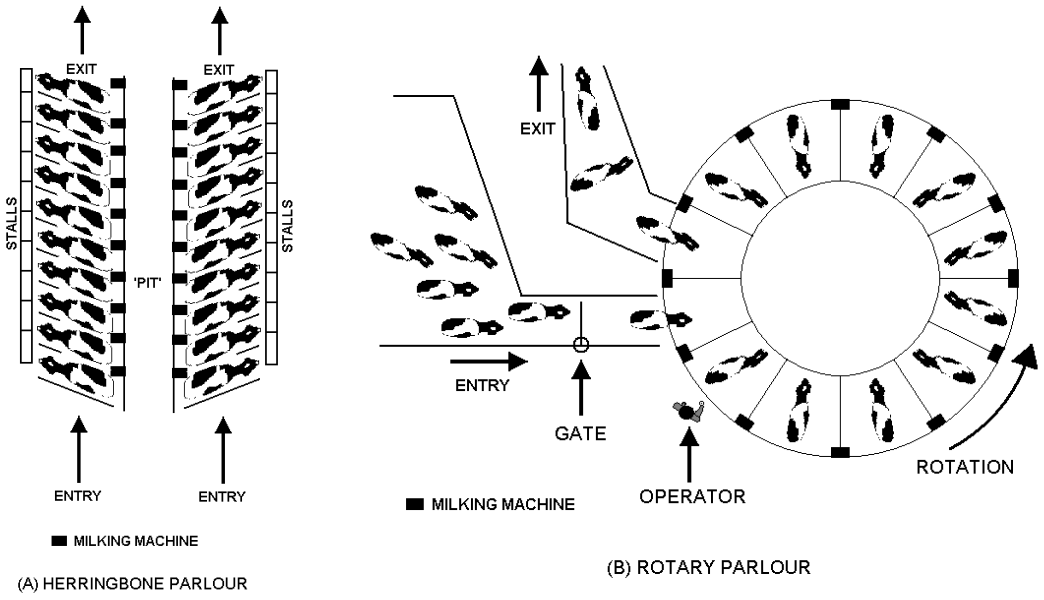


Figure 1.2: Common parlour configurations [9].

The rotary parlour has a higher capacity although is less common than the herringbone parlour. The idea is that the person applying the cups stays in the one spot and the cows, standing in stalls arranged radially on a raised turntable, are brought to the point at which the cups are applied. In this set-up, cows are constantly entering and exiting the parlour and the operator is constantly applying the cups. The rotary carousel parlour would typically handle up to 120 cows/hour with numbers up to 300 cows/hour possible in larger, well-managed systems. Studies have shown that the time taken to apply all four milking cups to the teats of a cow is approximately 20 seconds [5][10], however this can be as low as 10 seconds in working rotary parlours [9].

The objective of this project is to develop an actuation system capable of retrieving a set of milking cups from a given position and to place them on the teats of a cow. The system must be capable of taking three-dimensional coordinates from a potentially noisy source (most likely a vision system) and successfully applying the cups to the teats

reliably.

Furthermore, the actuation system must be suitable for use in a high milk-production scenario with as little modification to farming methods, routines or milking parlour equipment as possible. This means designing for use in a rotary milking parlour where access to the teats is only available from between the hind legs of the cow. The system must not create a delay in the existing procedure and thus must apply the cups at least as fast, if not faster, than a human operator.

In terms of coordinate systems, the base of the rotary parlour (carousel) will be fixed to the floor as will the base of the robot. These will thus not move relative to each other. A coordinate frame with its origin at the centre of the base of the robot will be used as the world coordinate frame, relative to which everything will ultimately be measured. Each stall on the carousel will rotate about the central axis of the parlour. As the cow stands within the stall, its body position will be restricted by what is known as a crush barrier. This allows a limited amount of movement of the cow. It may shift its weight when in the stall but will not be able to walk away. Similarly, each time a cow enters the stall it will be in a slightly different position but still very close to a mean position. Thus the position of the cow will vary relative to the stall but will have a constant mean over time. The variation will be seen as random noise about the mean. The teats also may change position relative to the cow as it moves in a short timeframe due to a possible swaying motion of the udder. These will be small movements and the teats will return to a mean position. Over a longer period of time, months or years, the teats will change position. They may move apart as the cow grows or sag together as it matures. This will also be connected to the lactation cycle of the cow. The teats will thus appear to have a velocity relative to the cow over long periods of time.

1.2 Development of Robotic Milking Systems

Several robotic milking systems are already currently on the market.

- RDS Futureline Mark II manufactured by SAC [11]
- DeLaval AMR manufactured by DeLaval [12]
- MIone by GEA Farm Solutions [13]
- Astronaut A4 by Lely [14]
- Merlin 225 by Fullwood [15]

However, several drawbacks make them unsuitable for use on the majority of Irish farms. Most of these existing systems have been designed for dairy farms where the animals are predominantly kept in sheds, as opposed to pasture-based farming. In these systems, the cow itself decides when milking occurs, hence the term voluntary milking system (VMS). In pasture based milking, the cows are taken from the fields to the milking parlour in batches to be milked. This can be performed up to three times a day. In a VMS, when the cow decides to be milked it moves to the stall passing a sensor that identifies the cow and allows it to pass through to the stall. The animals need to be screened for suitability to be milked as they may be on antibiotics, or it may be too soon after a previous milking etc. If the cow is suitable, it passes into the milking stall where the robot cleans the teats of the cow, scans their positions and applies the milking cups. The cow remains in the stall until it has finished giving milk at which point it is allowed to continue back to the general population area of the shed. See Figure 1.3.

This style of milking parlour is representative of a very different model of dairy farming to that which is used in Ireland. Shed-based farming is used more in countries such as the Netherlands where there is not such an abundance of land. Countries such as Ireland, the UK, the US, and New Zealand more commonly use the pasture-based farming model. While it would allow the use of robotic milking systems, converting to a shed-based system would be a very costly exercise. Since it is a goal of this project to interfere as little as possible with existing parlour equipment, changing the entire farming model is not an option.

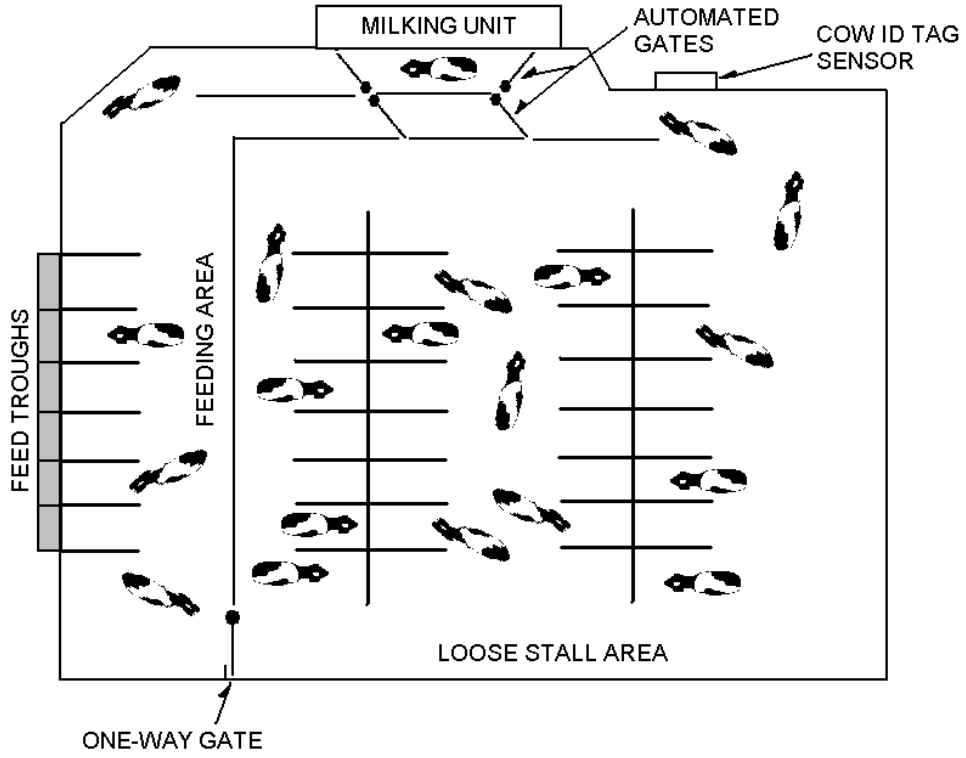


Figure 1.3: Layout of a shed-based voluntary milking system [9].

Another major issue with voluntary systems is the fact that once the cow has begun milking, the robot is not able to work on any other cow, with the exception of the SAC RDS Futureline which services two stalls at once. This is compared to a rotary parlour where anywhere from 10 up to 80 cows can be milked simultaneously. In these voluntary milking systems (VMS), time spent applying the cups to the teats accounts for a small proportion of the time spent milking each cow. As a result, the application itself is really not time critical and can take up to several minutes.

Available systems tend to apply the milking cups to the teats sequentially, that is, the first teat is detected and the cup is applied before moving onto the next teat. This allows a simpler end-effector design, in that only one cup needs to be positioned at a time, but the trade-off is that application takes far longer.

The exception in the list previously given is the DeLaval AMR, released in the last year which is the first robotic rotary milking parlour system. This is discussed in the next section.

1.3 Case Study: DeLaval AMR

The system with the highest throughput is the recently launched DeLaval AMR, billed as the worlds first automatic milking rotary system. The DeLaval AMR design is based on a rotary carousel milking parlour with up to five robots positioned inside the parlour, Figure 1.4. In this setup, some of the robots are preparing the teats and some are applying the milking cups. The robots are in fixed positions and the parlour rotates, bringing the cows to the robots. According to the published documentation [16], the system is capable of milking up to 90 cows/hour. The system is modular, meaning that additional robots can be added increasing the capacity to this maximum. This compares to 50-75 cows/hour in a herringbone parlour and 120-300 cows/hour in a rotary parlour. It is designed for herds of up to 300 cows. Figure 1.5 shows the system without cows present. The two robots closest to the camera have different end-effectors to the other two. These are designed to clean the teats, while the other two robots use magnets to hold two milking cups simultaneously.

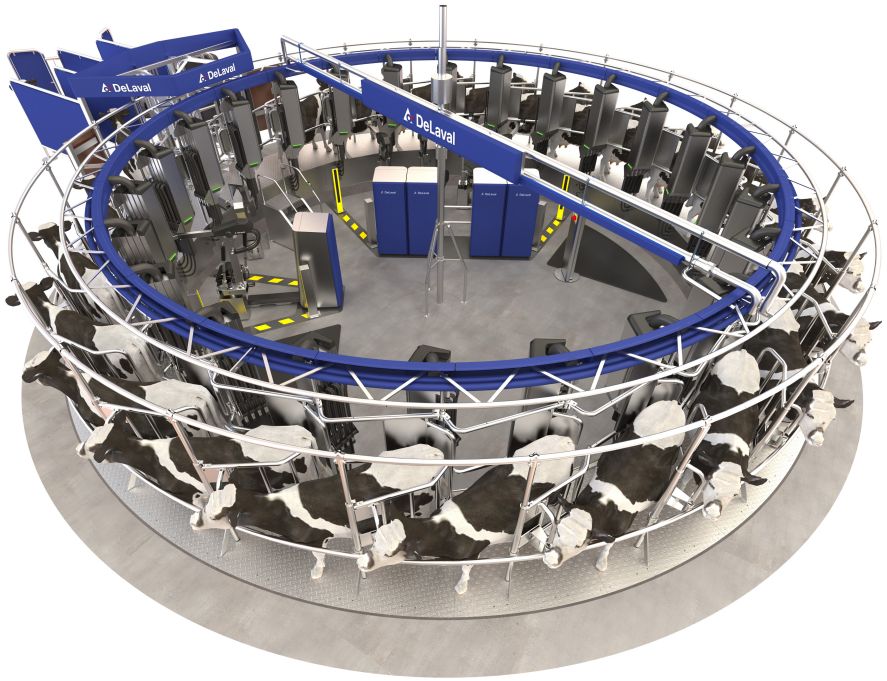


Figure 1.4: Overview of DeLaval AMR.

Figure 1.5: DeLaval AMR system showing four robots and no cows.

When the cow enters the system an identity tag is read calling up information on the cow, including teat positions. This information is used along with a laser scanner to find the positions of the teats to allow the preparation robot (of which there may be two) to clean, dry and strip (the removal of a small amount of milk discarded with the wash water) the teats. The carousel then progresses the cow to the application robot (again, there can be two of these). The robot retrieves the milking cups from a magazine mounted on the side of the stall. The application robots use magnetic grippers to hold two milking cups at a time. These are placed on the teats of the cow sequentially. The system uses the historic position data and then the laser scanner for more accurate positions. The application process takes 20-30 seconds to apply cups to two of the teats. The milk is monitored for various parameters and each cup is retracted into the magazine when the milk stops flowing.

Since this system uses two robots to prepare the teats and two to apply the milking cups, this means that four cows are being worked on simultaneously. A fifth robot is used after the cups have been removed to spray disinfectant on the teats.

The robots used in this system are custom designed robot arms that more closely resemble industrial robots than the bespoke arms seen in the various other milking robots. This reflects the need for more compact yet flexible actuation systems. This milking system, as with all others currently on the market, accesses the teats from the side. This makes it necessary to use a specially designed carousel in which the cows are arranged at an angle exposing the side of the cow. On a regular rotary carousel, the cows are in parallel stalls. This means that although the DeLaval AMR is suitable for use in pasture based farming, the milking parlour will still need considerable renovation. The throughput of the system is also very low compared to a regular carousel parlour where cup application is done by hand. This system can milk up to 90 cows/hour but the manual system can be up to 300 cows/hour. For a system that uses five robots this does not seem very efficient. Since the robot takes up to a minute to apply the four cups, multiple robots are necessary to bring the throughput up. In comparison, a human can apply the four cups in about 10 seconds.

1.4 Main Challenges Facing Current Robotic Milking Systems

As can be seen from a study of an existing system (Section 1.3), there are several main areas where improvements must be made in robotic milking in order for it to be successful in a pasture-based dairy farming model.

1.4.1 Workspace

All systems currently on the market assume side access to the teats of the cow. This is sufficient in a shed-based voluntary milking system but not for use in either a herringbone or a rotary parlour. Where it is used in a rotary parlour, the carousel must be completely redesigned and since the cows are sideways, fewer can fit on the carousel at a time. In

order to access the teats from between the hind legs of the cow a smaller more dextrous robot arm and end-effector must be used.

Using data from [17], as well as from measurements made at a working rotary parlour in [9], the largest, smallest and average dimensions likely to be found on a dairy cow can be modelled. This is discussed in Section 2.1. These dimensions can be used to determine the necessary dimensions of a bespoke robot arm/end-effector.

1.4.2 Speed

In voluntary milking systems the time taken to apply the cups to the teats of the cow is not a major concern. The milking of the herd is spread out over a 24 hour period so the system can apportion as much time as is necessary when milking each animal. As a result, the systems can take the time to perform accurate laser scanning of the teat area. This is also used in the DeLaval AMR with the result of slowing down the milking process compared to manual application. While it is a reliable method of getting the cups to the teats, it makes the system unsuitable in the majority of high production farms. An alternative would be to use less accurate sensors and to incorporate filtering and mapping techniques to reduce positional errors. Since a robot arm can move faster than a human hand, the time savings in the system must come from the part of the system that deals with finding the coordinates of the teats. Gains can also be made using simultaneous application.

1.4.3 Sensing

Existing systems generally use lasers to find the coordinates of the teats. In this method a line of laser light is scanned across the udder and a camera detects the line. The system then builds a three dimensional map of the area by analysing the contours of the line. This process of scanning the teats can take several seconds per teat. When this method of sensing is used to detect the teats in a sequential manner, it takes significantly longer

to apply the four cups.

An alternative solution would be a sensing system that does not require the sensor to be moved about. Ultrasonic sensors need to be positioned beneath the teats in order to measure the teat positions [18]. This is not suitable since a rough idea of where the teats are located would be needed in the first place. It also would require a certain amount of time to place the sensor in position and scan the udder area and retract it again to allow the cups to be placed.

A better sensing system would use cameras, possibly including thermal vision, that remain stationary. Additionally, sensing would be less time consuming if all four teats are found simultaneously.

1.5 Proposed Solution

Having outlined the reasons for the unsuitability of the robotic milking systems already on the market, several criteria will now be outlined that are deemed necessary to develop a system capable of applying milking cups to a cow on a standard rotary parlour.

1.5.1 Simultaneous Sensing

When applying the milking cups, sensing incurs a large time overhead in the process in existing systems. One of the reasons for this beyond the method of sensing used is the individual scanning of the teats. This effectively multiplies the sensing time fourfold. A time-critical system would need to detect all four teats simultaneously in order to avoid the process of scanning, applying, scanning, applying etc. for all four teats. If the four teats were captured in one image or in each of a stereo pair, the coordinates of the four teats could be found together in one go. This would drastically cut down on the time taken to apply the cups [19].

1.5.2 Simultaneous Application

Another way of cutting down the time spent on each cow would be to bring the four cups to the teats at the same time. Existing systems retrieve a cup then place it then return to retrieve another cup to be placed or at best they do this with two cups at a time. This sequential method of applying the cups is essentially four (or two in the latter case) times slower than it needs to be. With an end-effector capable of handling and positioning four cups at the same time, it will be possible to significantly reduce cup application time [9].

1.5.3 'Mothership' Approach

Combining simultaneous sensing and simultaneous application, we get a system where the sensors (most likely a vision system) identify the four teats of the cow in the stall by taking a set of images simultaneously (e.g. a stereo pair) and calculating the coordinates of the ends of the teats relative to the base of the robot. A vector will be added to this to account for the movement of the parlour. While the vision system is finding the teats, the robot arm actuator retrieves the four milking cups from a magazine or rack and moves into a neutral position ready to move the end-effector under the cow. When the coordinates have been calculated the robot arm moves the cluster of cups to a position beneath the udder. The four cups, manipulated independently in the horizontal plane by the end effector will be positioned underneath each teat. The arm will then rise a small amount to place the cups, release them and retract to a neutral position. The process will start again as the next stall on the carousel moves into position.

1.5.4 Design Challenges

The first design hurdle that must be crossed is to conceive an end-effector that is capable of carrying four milking cups simultaneously and moving them into the most extreme configurations likely to be seen on a dairy cow. This means the end-effector must be able to place the cups on a cow whose teats are either very far apart, very close or a

combination of the two for various teats. The end-effector must be compact enough so it can fit between the hind legs of the smallest cow likely to be seen but still have the reach to apply the cups to the largest.

One of the biggest challenges of this project is the design of an autonomous sensing system capable of finding the teats with sufficient accuracy in a sufficiently short time. Work has already been done in this area on this project [19]. It was found that a stereo vision system combined with a thermal camera could provide the necessary level of accuracy in a comparatively short space of time. This however requires more development to improve teat recognition algorithms and is beyond the scope of this thesis.

Since the sensing system may have limited accuracy, the possibility exists of using historical data to reduce errors in the measured coordinates. Assuming the teats are always in the same place is really not feasible as there are many factors, biological and otherwise, that cause the teats to be in different positions globally and relative to one another. Instead, the data from current measurements can be combined with data from previous milkings to give an improved estimate of the actual positions of the teats. With this approach it is hoped that an efficient, high speed and yet robust system will result.

1.5.5 Accuracy

Early in the project, a goal of ±5mm was identified as a sufficient level of accuracy to ensure that the cups are successfully placed on the teats. The reason such a seemingly large error is allowed is because of the pliability of the teats themselves. The vacuum coming from the milking cup that holds the cup in place on the teat, pulls the teat in if it is sufficiently close. Discussion with farm workers established that this happens up to a distance of around 5mm.

1.6 Literature Survey

Any robot that interacts with its environment needs to be able to perceive that environment. For the sake of this discussion we will assume the ability of the robot to move about in its environment. Perception is the concept of the robot using sensors to interpret the world around it to allow it to perform tasks. This interpretation can take the form of a model or a map of the world. This model can be updated by incorporating measurements taken by the sensors and can change over time. Part of the problem of creating such a map originates with the sensors used. The sensors of a system cannot provide completely accurate data on the entire environment due to stochastic noise and aliasing. An accurate map is therefore not available. Estimation is how a system interprets these noisy sensor values as beliefs of the state of the robot and/or its environment. The other side of the problem is that these sensor errors prevent the robot from recognising features on the map and thus prevent it from accurately finding its own position in the environment. A further discussion of perception and estimation can be found in [20].

1.6.1 Estimation

To estimate the true value of a variable measured by a sensor, several data estimation techniques can be used, specifically, the Kalman and particle filters. Both of these are implementations of the Bayes filter. This algorithm combines a model of the system with measurement values to provide an improved estimate of the state of a variable. This is discussed further in Section 4.3. Discussions can also be found in [21] and in [22].

1.6.1.1 Kalman Filter

The Kalman filter was first introduced by [23] and [24]. It was originally developed for use in aerospace applications - Kalman's original work was part funded by the US Air Force [24] - but has since been used in diverse areas such as robot navigation [25], analyzing census information [26], modelling stock returns [27], predicting energy spot prices [28],

spacecraft tracking and flood prediction [29]. An in depth study of the Kalman filter, including its continuous-time form (Kalman-Bucy filter) and the extended Kalman filter for non-linear systems can be found in [29].

1.6.1.2 Particle Filter

The Kalman filter is a parametric implementation of the Bayes filter. This means the belief of the current state of the system generated by it is represented as a Gaussian distribution with the parameters being the mean and the covariance. The particle filter on the other hand is a non-parametric implementation, since it represents the belief with a set of estimates known as particles. The most influential paper on particle filters is [30] in which it is referred to as a bootstrap filter. The roots of the particle filter go back to [31]. This filter takes a range of samples or particles from a belief distribution and propagating them through the system equations. Each is given a weight and the particles are resampled to give a new set of estimates. The particle filter, sometimes referred to as sequential importance sampling or sequential Monte Carlo filtering, is described in [32], [33] and is the subject of [34]. A tutorial is also given in [35]. A comparison of resampling methods is performed in [36].

1.6.2 Stereo Vision

Two of the most standard texts in the area of stereo vision are [37] and [38]. These are referenced throughout the discussion on stereo vision theory, Section 3.1. All of the theory behind the camera models developed herein can be found in these sources. The camera calibration procedure itself makes use of the Camera Calibration Toolbox for Matlab, developed by Jean-Yves Bouguet at Caltech [39]. This is based on the camera model described in [40]. This is essentially the same as the model given in [37] with the addition of a lens distortion model. The particular distortion model accounts for radial and tangential errors and is also known as the plumb line model. This was first put forward in [41].

Chapter 2

Actuation System

The actuation hardware and its associated control hardware was designed and specified in earlier work [9]. At the onset of the current phase in research, the robotic system was in place with low level control of actuators but no combined system control was available.

2.1 Hardware Description

2.1.1 Workspace and biometric considerations

The robot workspace is defined by both mechanical elements (guard rail height, stall width etc.) as well as biological (width between legs of the cow, spacing of the cow's teats etc.). The most important dimensions of the stall are shown in Figure 2.1. These can be obtained from the manufacturer's design specifications.

In order to design for this environment, biometric data is used. This data gives information about the average, maximum and minimum dimensions of certain characteristics of the animal. Information in two areas is of particular interest: the space between the rear legs of the cow and below the udder and also the size and relative positions of the teats themselves. Biometric data on teat positions has previously been published [17] but not for spacing of the hooves and hocks. This data was gathered at the Teagasc Moorepark Agricultural Research facility in Fermoy, Co. Cork [9].

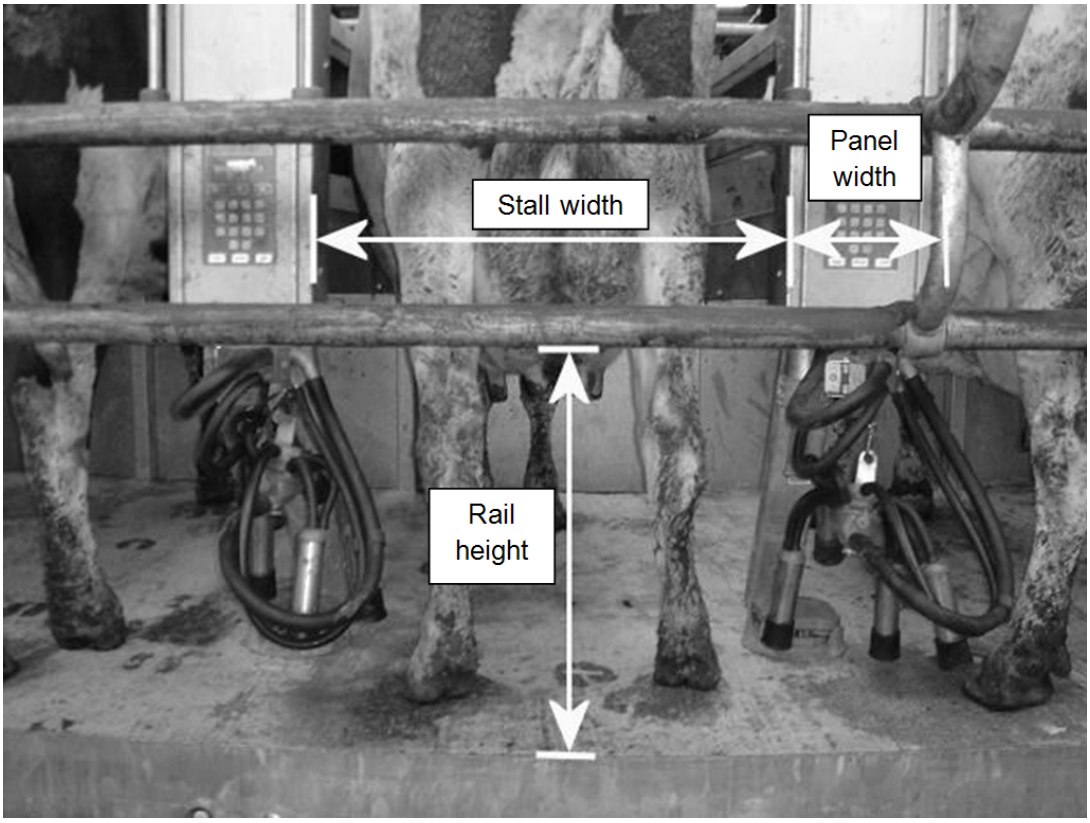


Figure 2.1: Major dimensions of the rotary parlour stall [9].

Since the main goal is to create a system capable of access from between the rear legs of the cow, the end-effector and related service lines (power, control etc.) must be able to fit into the prescribed space. This space is defined by the height of the teats of the cow from the floor of the stall and the distance between the legs of the cow. Figure 2.2 shows the maximum, minimum and average workspace volumes beneath the cow, as calculated from the data obtained at Moorepark, summarised in Table 2.1.

Table 2.1: Biometric data from 34 worst-case representatives of a herd of 200 cows [17]

Trait	Mean	Standard Deviation	Minimum	Maximum
Width at hocks (mm)	184	40	115	256
Width at hooves (mm)	216	58	115	333
Floor to lowest rear teat tip (mm)	426	63	291	552
Distance from rear teat to rail (mm)				250

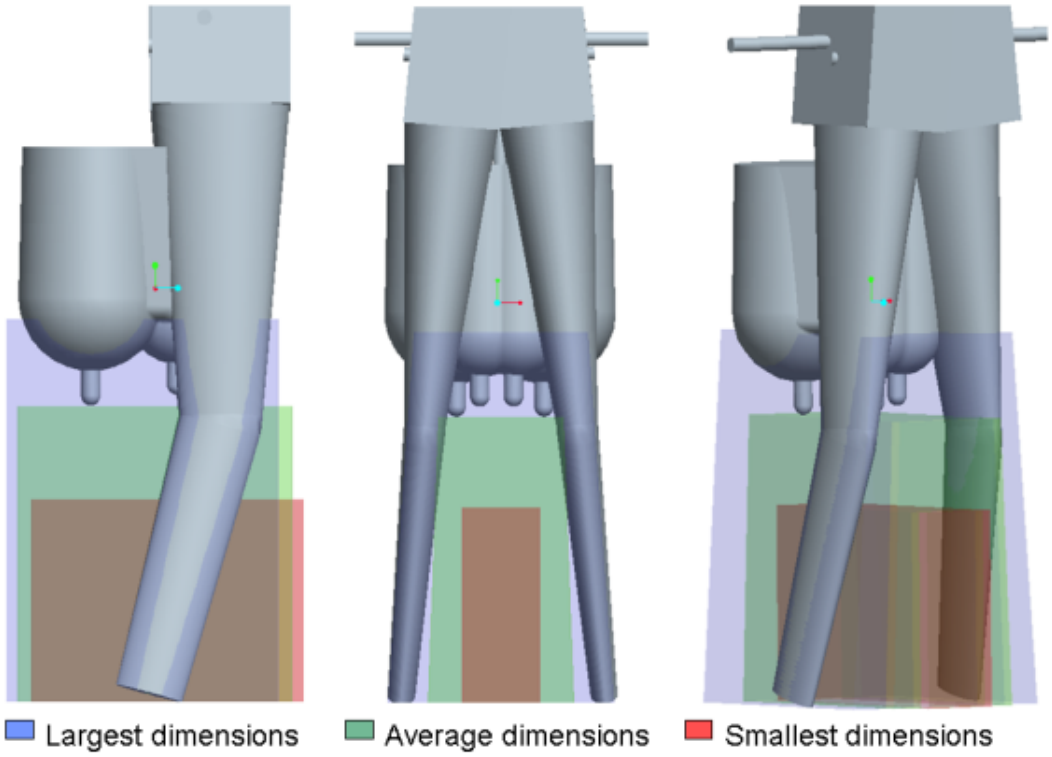


Figure 2.2: ProEngineer model of workspace reflecting data in Table 2.1 [9].

2.1.2 Robot

The largest component of the actuation system is the 6-axis robot to which the end-effector is mounted. This provides the reach to move the end-effector from a home position, clear of the teats, to the stand on which the milking cups are mounted and on to the workspace of the teats. The alternative to using an off-the-shelf robot is to design a custom robot arm. This has been done by [14] and [15] but does provide a whole set of new design challenges. In this application a lot more flexibility is required - the kind of flexibility provided by standard industrial 6-axis robots. This saves on a major design stage and allows resources to be focussed on other aspects of the system. The idea of using a standard robot arm in a milking system has been used by [11], although in a considerably different configuration to that used here.

The robot used in this project, a Nachi SC35F-01, is shown in Figure 2.3. This can handle a payload of up to 35kg, higher than needed. It also has a reach of about 2000mm,

depending on the joint configuration. It also has the accuracy needed to move the end-effector to the best position to allow it to place the cups. The coordinate frame used by the robots own controller is also shown at the base of the robot in Figure 2.3. When the robot is sent to a certain point, the coordinates given to it are the coordinates of the centre of the toolplate in this frame. This frame is used throughout the project as the world coordinate frame and will be referred to frequently in this work.

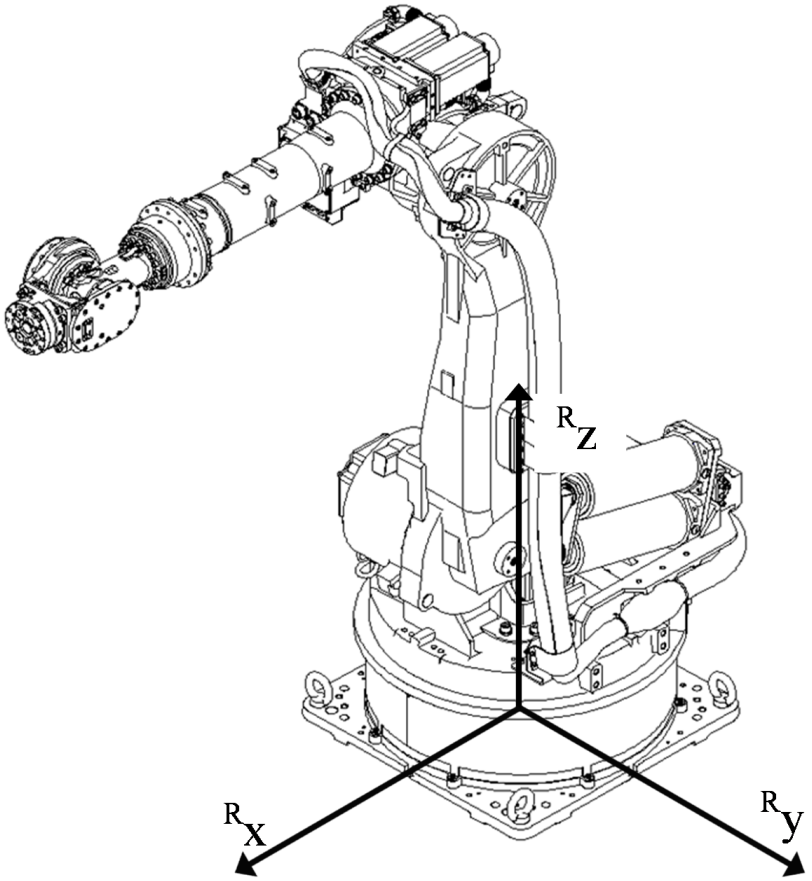


Figure 2.3: Nachi SC35F 6-axis robot and robot or world coordinate frame.

2.1.3 End-effector design

In order to reduce the complexity of the end-effector, it was assumed that the pliability of the teat would mean that it would not be necessary to adjust to the vertical angle of the teat i.e. the suction from the milking cup would pull the teat into it. The data in [17] shows that the difference in height from the floor between the front and rear teats

of a cow are about 10mm for the smallest and about the same for the largest animals. On an end-effector without individual height adjustment for the cups, it can be assumed that the four teats are at the same height. The end-effector can move into position at the lowest height and move vertically several centimetres to ensure the cup goes onto the teat. This means a difference in height of 10mm would be negligible.

As Figure 2.4 shows, the end-effector consists of a chassis that supports four smaller arms. Each arm has two degrees of freedom in the form of a linear movement in the ± x-direction and a rotational movement about an axis parallel to the robot z-axis (i.e. vertical). The rotational, or revolute, actuators are positioned on the end of the linear arms. A diagram of the area covered by this configuration is shown in Figure 2.5. Since the revolute axes would be in the most confined area of the system i.e. directly under the teats, servomotors were chosen. These are available in very compact packages that include the motor, the gearbox and the position encoder in less space than a stepper would take. Although the reduced size and mass were a major advantage, the backlash resulting from the use of the planetary gearbox was expected to result in a slight loss in accuracy.

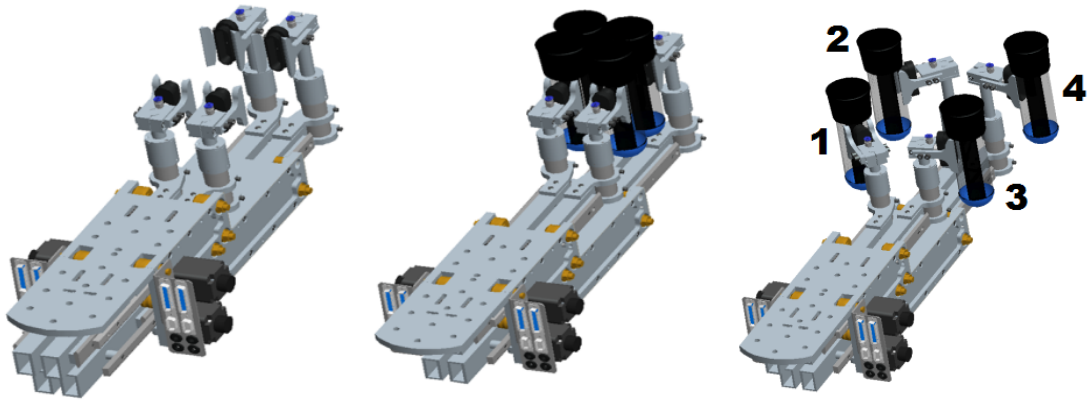


Figure 2.4: ProEngineer model of the end-effector with and without cups, and in various cup configurations. The teat numbering convention used throughout the project is also shown.

The linear actuators use rack gears to move the 'forearm'. The motors used are stepper motors. These were chosen due to their positive braking action. The larger

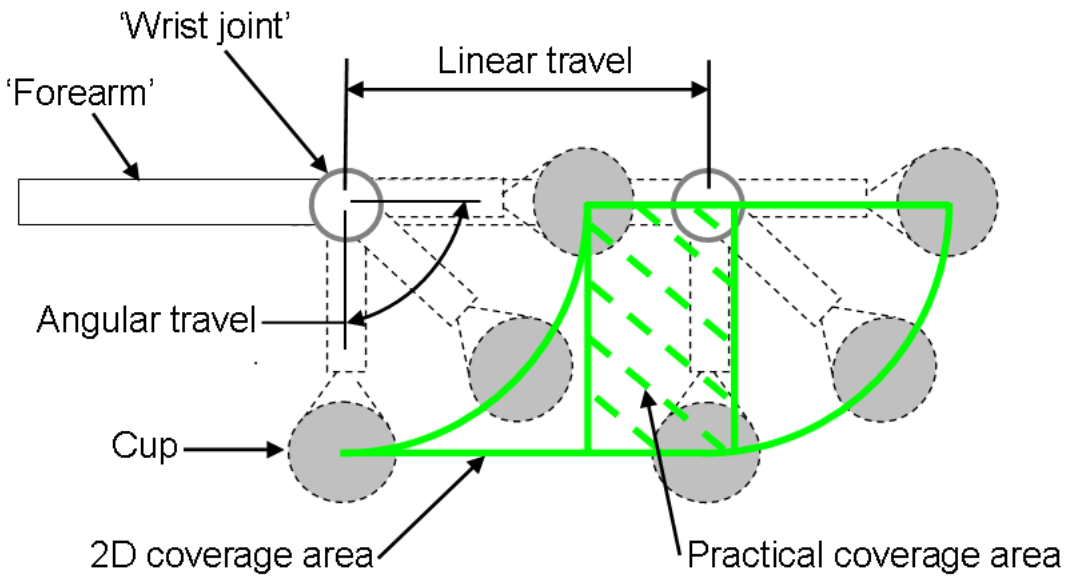


Figure 2.5: 2D area covered using the linear/revolute mechanism (top view).

inertial mass of the linear arm served to damp the vibrations associated with stepper motors. Stepper motors and their associated drive hardware are also considerably less expensive than servomotors of a similar size. The steppers were mounted laterally on the chassis of the end-effector. Although the large size of the stepper motors was a drawback, they were placed back far enough that the overall width at this point would not be an issue. A photo of the end-effector, not attached to the robot, is shown in Figure 2.6

2.1.4 Control Hardware

The servomotors were purchased with matched controllers for the particular motors and encoders. These hardware units are called EPOS 24/1 units (manufactured by Maxon Motor) and were mounted in the elbow housing on the robot. These units use PID position control with a self-tuning routine to find the regulator gains. The EPOS units come with proprietary software to allow for the configuration and use of the motors. Library files are also supplied allowing the various functions of the software to be called from the Labview environment without running the software itself. Communication with the EPOS units is via 3-wire RS-232 cable. Each revolute axis also has a home position sensor. These are photoreflective sensors that emit and detect infrared light. They are

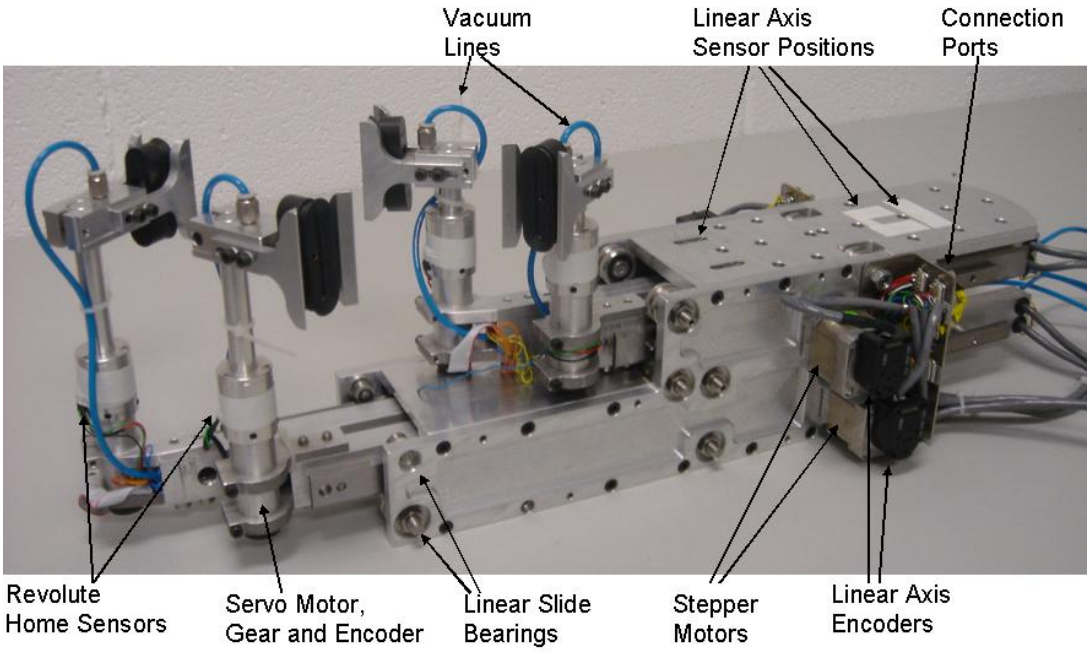


Figure 2.6: End-effector - not attached to robot arm.

positioned to detect the non-reflective grub screw that holds the revolute axis shaft to the shaft of the servomotor. This sensor is fed back to the EPOS control unit.

In order to control the stepper motors a National Instrument motion controller PCI card is used. This card generates step and direction signals for the motors. The card (and hence the motor control) is configured using National Instruments own software, Measurement and Automation Explorer (MAX). This allows great flexibility in how the motors are run (e.g. open / closed loop, step & direction / CW & CCW pulses etc.). A huge advantage in using the National Instruments products is the high compatibility - the motion controller card is easily configured and controlled through Labview which is itself a National Instruments product. The step and direction signals are routed from the PCI card through a breakout box to the stepper drives. These drives convert the input signals to correctly phased drive currents for the steppers. As with the servomotors, the stepper motors have encoders providing positional feedback to the control system. These are routed back to the PCI card via the breakout box. In addition to this, the linear axes each have three sensors along the length of movement. These include two mechanical limit switches activated by tabs bonded to the forearm and an optical switch at the home

position. Mechanical switches are used for the limits due to lower cost and lower accuracy required. The home position is more critical so optical switches are used. These home and limit switches are fed back to the PCI card also.

The robot arm ships with its own hardware controller. This runs its own operating software on a version of Microsoft Windows. As a standalone unit, it can be programmed to run complex sequences of movements. However it is also possible to transfer pose variables (a set of position values for each joint of the robot) via a serial cable from the PC. The PCI motion control card, used to control the stepper motors, also has 32 lines of digital I/O that can be used to send and receive control and status signals directly to and from the I/O board of the AX controller (albeit via a breakout board). Again, this signalling is all controlled from Labview. This precludes the need for additional RS-232 ports on the control PC.

The relationship of elements of the hardware control system are summarised in Figure 2.7.

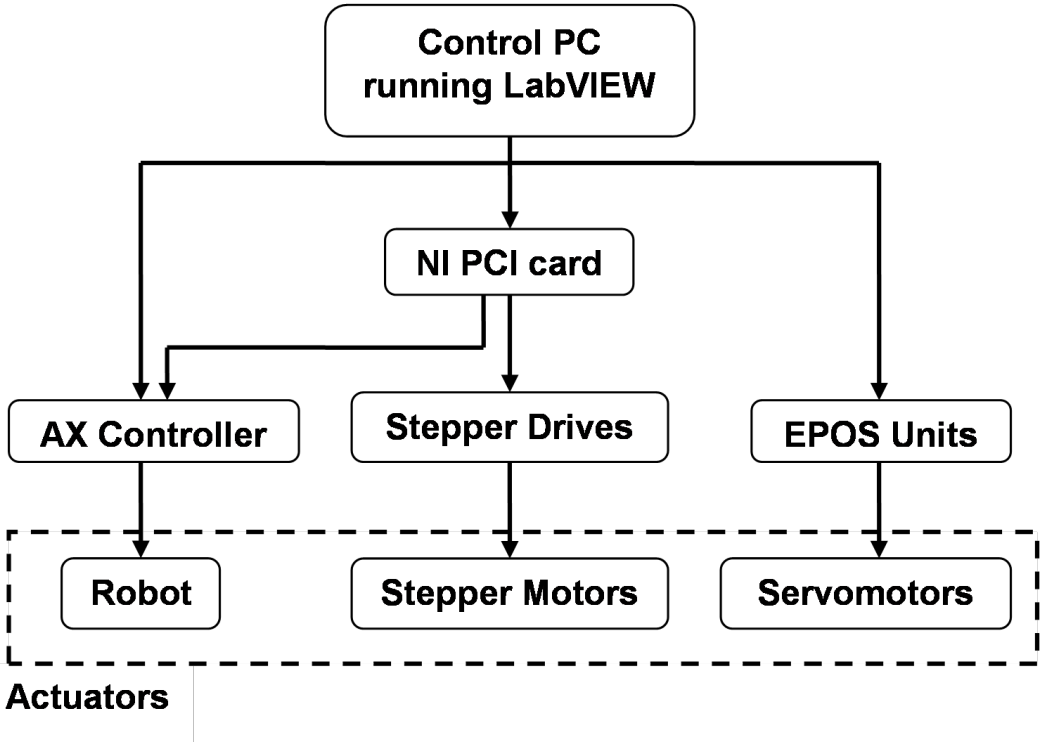


Figure 2.7: Flow chart of control hardware for actuation.

2.1.5 Microscribe

One piece of equipment that is integral to the project if not the system itself is the Microscribe measurement probe, seen in Figure 2.8. This was used as a ground truth in testing, for measuring teat positions during development and measuring dimensions of various parts. The probe has five joints in series, similar to the robot arm. There are no motors but position encoders allow the proprietary software to calculate the three-dimensional coordinates of the tip of the probe when the footswitch is pressed and output them to a range of file types. The coordinates are returned in a frame set up automatically when the probe is switched on but the software allows for a new, known coordinate frame to be set up.

Figure 2.8: Microscribe measurement probe shown mounted to a steel plate with footswitch.

To set up this coordinate frame, three points on its x-y plane are selected using the probe. The origin of the frame is then selected followed by a point anywhere along the x-axis and another on the y-axis. Initial attempts to use the floor of the lab as the x-y plane failed due to the difficulty in marking out reliable orthogonal x- and y-axes. Instead, the robot itself was used. A point was marked on the end-effector at a known offset from the centre of the toolplate. The end-effector was then moved to a series of positions and the marked point selected by the Microscribe each time. In this way a coordinate frame was set up with axes parallel to those of the robot coordinate frame. Coordinates found in the Microscribe frame thus needed only to be translated and not rotated to be transformed into robot coordinates.

2.1.6 Teat Rig & Milking Cup Stand

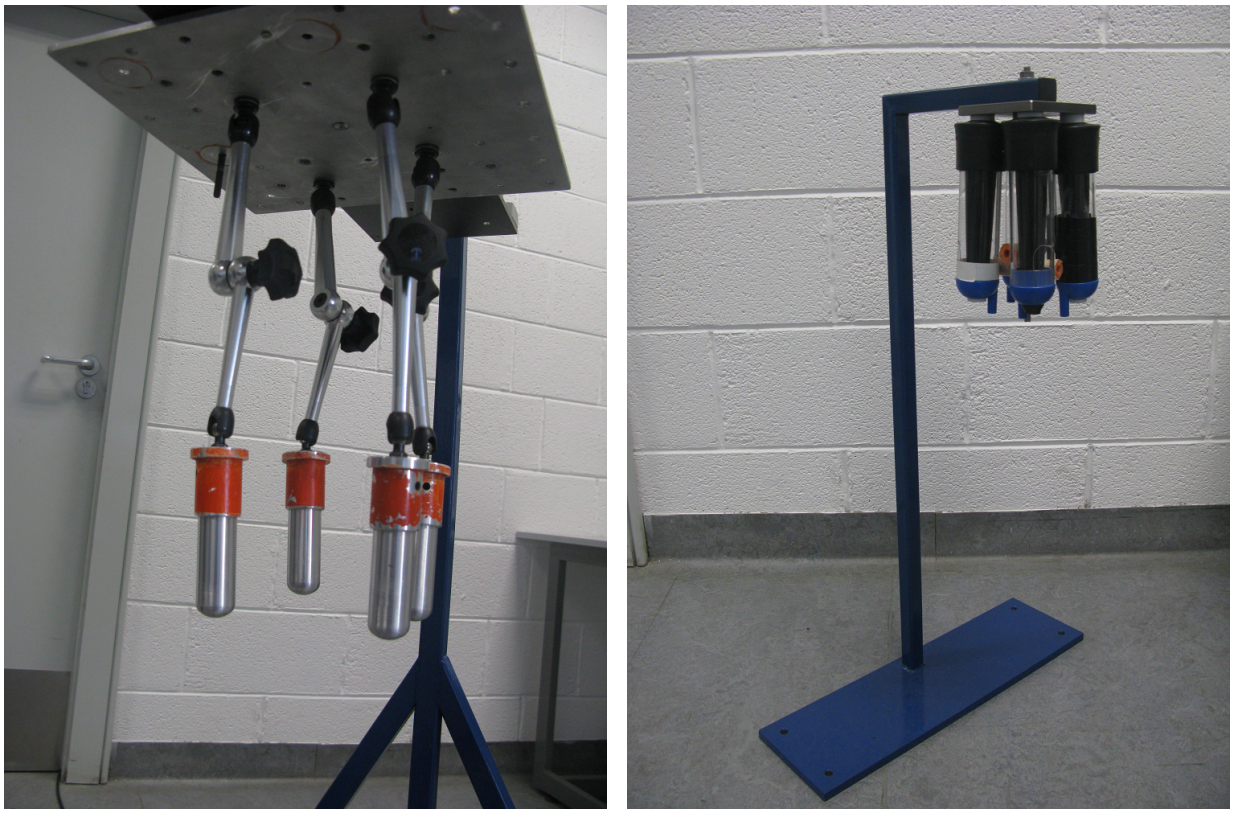
At this stage in development, testing was performed on a set of dummy teats. Live animal trials will come later when the system has been developed to be reliable and accurate enough to be safe. The dummy teats were machined from aluminium and mounted by universal joints to an armature. These joints allowed the teats to be positioned freely in space and to hold their positions. The teats are shown in Figure 2.9a.

This rig was later modified to allow the four teats to be moved in unison in a straight line. The plate to which the teats are attached was mounted to a screw-adjusted slider bolted to the frame of the teat rig. This is shown later in Section 5.5 in Figure 5.9.

A stand was also manufactured to hold the four milking cups. This was located near the teat rig and enabled the robot to retrieve the cups to bring them to the teats. This is shown in Figure 2.9b.

2.2 Control of Hardware

The hardware in the system is controlled from a central PC running Labview. This allows coordinated movement of the entire actuation system based on information received from



(a) Dummy Teat Rig (b) Milking Cup Stand

Figure 2.9: Photographs of the dummy teats used as targets for the milking cups and the stand upon which the milking cups were stored.

the vision system. Labview allows for straightforward interfacing with the various control elements of the system particularly the National Instruments PCI card used to interface with the steppers and the AX controller.

2.2.1 'Mothership' Approach

The end-effector was designed to apply the milking cups in parallel to the teats. The idea is that all four cups are held by the robot simultaneously. They are taken under the cow and positioned at the same time. They are released together and the robot withdraws. This is crucial to achieve one of the objectives of the project: that of creating a system capable of placing the cups as fast if not faster than a human operator.

When the actuation system control software receives the coordinates of the teats, the average of the four coordinates is found. The centre of the cluster of cups is then placed

directly below this point. The angle of the centreline of the teats (a line joining the midpoint of the front two teats to the midpoint of the rear two teats) is also calculated. This is so that if the cow is standing at an angle to the robot, the end-effector can be re-oriented. This means less extreme positioning for the end-effector actuators.

The actuator positions of the end-effector are then calculated individually from the kinematic model of the robot. This is discussed in more depth in Section 2.2.2. The cups are thus positioned under each teat and the robot raises the end-effector, applying the cups to the teats. The end-effector does not have the ability to individually adjust the heights of each cup, but since there is a difference in average heights of the front and rear teats of only 4mm [17] a vertical move in unison is deemed sufficient. The vacuum pads that hold the cups on the end-effector are then released and the robot retracts to a neutral position. This algorithm is illustrated in Figure 2.10. It is assumed that the cow will be in a relaxed state during the vertical motion of the application and will not move suddenly. If the cow does move unexpectedly during the procedure, an additional sensor monitoring the cow could cause an interrupt that would abort the application.

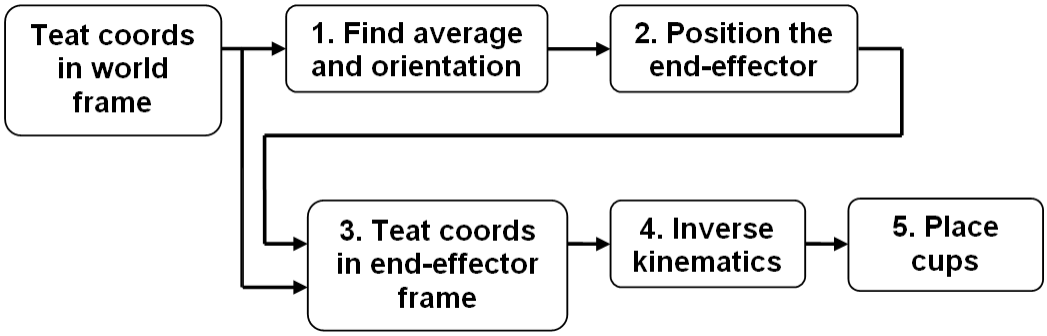


Figure 2.10: Flow chart of control tasks for retrieving/placing cups

This approach of bringing the cups to a central position and then individually positioning them is referred to as a 'mothership' approach. An alternative would be to position the cups sequentially, moving the entire end-effector each time. However this is less efficient, taking more time. Figure 2.11 shows the Labview block diagram of the mothership routine. The highlighted parts of the diagram correspond to the numbered

blocks in Figure 2.10.

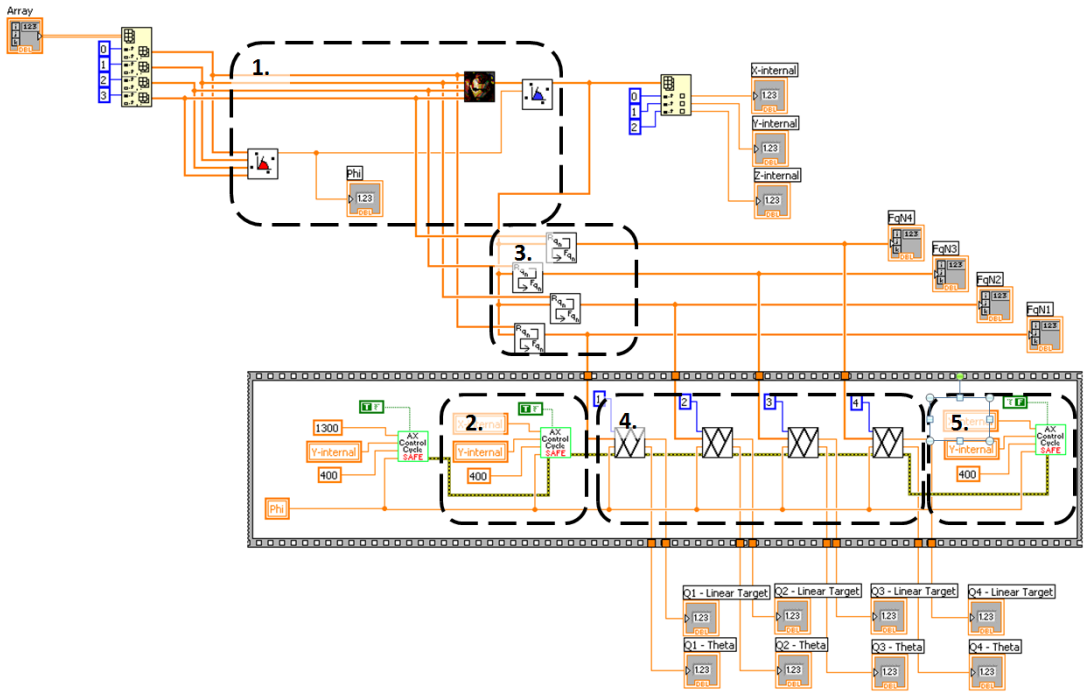


Figure 2.11: Labview block diagram showing 'mothership' method of moving milking cups to teat positions.

A variation of the block diagram seen in Figure 2.11 using the same principles was created to retrieve the milking cups from the stand. This moves the robot and end-effector to a pre-determined set of positions. At the point when the end-effector is in contact with the cups, the vacuum is turned on and the robot takes the cups to a home position. They are then taken to the teats using the above Labview programme.

2.2.2 Inverse Kinematics

The commands for the motors in the end-effector are calculated using a technique known as inverse kinematic analysis. In a kinematic analysis, the known lengths of the links in the mechanical system, as well as the current angles of joints are used to calculate the current position of the endpoint of the arm. In an inverse kinematic analysis, the desired endpoint is known and the required joint angles to reach this position are calculated. The kinematic analysis of one of the arms on the end-effector is shown in Figure 2.12.

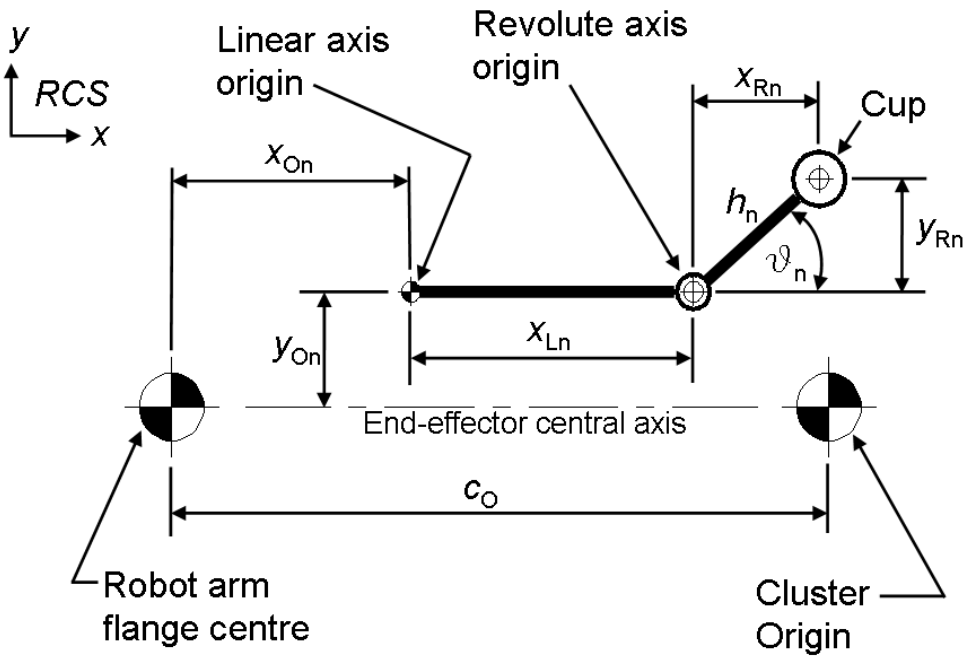


Figure 2.12: End-effector kinematics for one arm (top view) [9]

In this diagram the variables are:

x_{Ln} = linear axis displacement

θ_n = angle of the revolute axis relative to the tool central axis

x_{Rn} = x-component due to angle θ_n

y_{Rn} = y-component due to angle θ_n

The constants are:

x_{On} = constant offset in the x-direction from the flange centre

y_{On} = constant offset in the y-direction from the flange centre

h_n = length of revolute arm (from revolute axis to cup centre)

c_O = cluster origin offset from robot arm flange centre

The cluster origin shown is the geometric centre of the four cups when the end-effector actuators are in their neutral home positions. This point is positioned below the average coordinates of the four teats as returned by the vision system.

To find the equations that give the required joint variables we first need to establish a coordinate frame. In this case, the origin is taken as being the centre of the robot arm tool-plate. The axes of this frame are kept parallel to those of the robot coordinate frame. We then work from the base and find the transformation matrices that represent each translation and rotation necessary to relocate a coordinate frame from the base to the endpoint. This technique is discussed in more detail in [42].

The entire end-effector can rotate around the z-axis at the tool-plate with an angle designated ϕ. This is a constant calculated from the positions of the teats. This transformation is represented as $Rot(z, \phi)$. Next, there is a displacement along the new (re-oriented) x-axis by a distance equal to the constant offset x_{On} and the variable x_{Ln}. There is also an offset in the y-direction of y_{On} and no rotation. This is shown as $Trans(x_{On} + x_{Ln}, y_{On}, 0)$. At the next joint, there is a rotation about the z-axis by a value designated θ. This is the revolute axis and is described by $Rot(z, \theta)$. Finally the end-point is located a distance of h along the x-axis, thus $Trans(x, h)$.

These transformation matrices are combined by multiplying them in the order they occur, thus:

$$^0T_1 = Rot(z, \phi)Trans(x_{On} + x_{Ln}, y_{On}, 0)Rot(z, \theta)Trans(x, h) \tag{2.1}$$

This yields a matrix equation in terms of the known constant values of the end-effector. The only unknown values are x_{Ln} and θ, which are the joint variables. By setting this matrix expression to equal the desired position of the end-point i.e. the teat, we are able to re-arrange the expressions to give the following, where the desired end-point position is given by (p_x, p_y).

$$x_{Ln} = \frac{p_x - h\cos\phi\cos\theta - x_{On}\cos\phi + h\sin\phi\sin\theta + y_{On}\sin\phi}{\cos\theta} \tag{2.2}$$

$$\theta = \sin^{-1}\frac{p_y\cos\phi - p_x\sin\phi - y_{On}}{h} \tag{2.3}$$

These equations are implemented in Labview and the results are sent to the motor control elements of the software as position commands. The kinematic constants were measured using the Microscribe to improve accuracy.

2.2.3 Software Control of Actuators

2.2.3.1 Servomotors

The servomotors can be directly controlled using the manufacturer's own programme. This is useful for the setup and configuration of the motor controller units (EPOS) and also in troubleshooting. The controllers also ship with a package of .dll library files. These libraries contain functions for configuring and running the motors. They can be called from Labview meaning the stand-alone Maxon software does not need to be run in normal operation.

The motor can be initialised, moved and communication with it closed by calling the various functions found in the .dll library file. Labview has a Call Library Function Node that allows us to reference a library and call a function from that library. These functions include:

VCS_OpenDevice - opens communication with the EPOS unit

VCS_SetProtocolStackSettings - sends configuration parameters to the unit

VCS_MoveToPosition - sends the motor to a position which is sent in the form of encoder counts as a variable passed into the function

An example of VCS_MoveToPosition being called can be seen in Figure 2.13.

2.2.3.2 Stepper Motors

National Instruments' Labview software has a motion control toolbox available that allows control of the motors directly from with the Labview environment. The stepper motors can be configured and controlled from the NI Measurement and Automation (MAX)

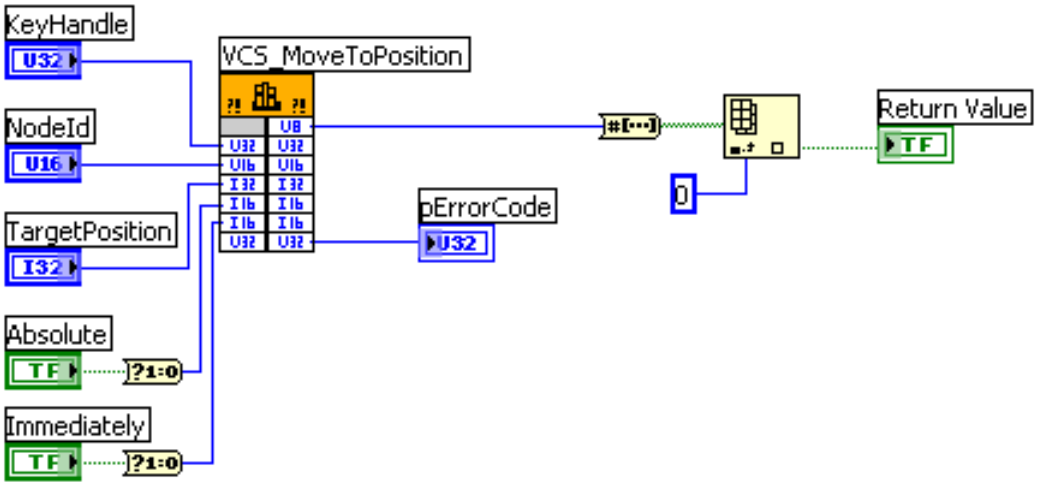


Figure 2.13: VCS_MoveToPosition called from Labview to move a servomotor.

software which interfaces the stepper motors through the NI PCI card. Again, this is useful for initially configuring the motors and for troubleshooting, but in the day-to-day running of the system it is only necessary to use the MAX software for initialising the PCI card. One of the limits of the PCI card is its ability to control only 4 motors. It was more suited to the stepper motors due the relative simplicity of the servo control.

The Motion Control toolbox contains Labview function blocks that load the velocity, the acceleration, the deceleration and the curve between acceleration / deceleration and constant velocity. It is thus possible to describe detailed velocity time curves for the motor to follow. As with the servomotors, the error flow in Labview is used throughout control of the stepper motors to allow easy identification of problems. The Labview block diagram to move a stepper motor, including setting velocity, acceleration and deceleration is shown in Figure 2.14.

2.2.3.3 Six-Axis Robot

A program was written for the industrial robot controller that is initiated by the Labview software. The program, when called waits for a general purpose input flag from the PCI card. This flag is set by the Labview software. Once the flag is received, the AX program sends a request for the pose variable and again enters a wait cycle. This data is received

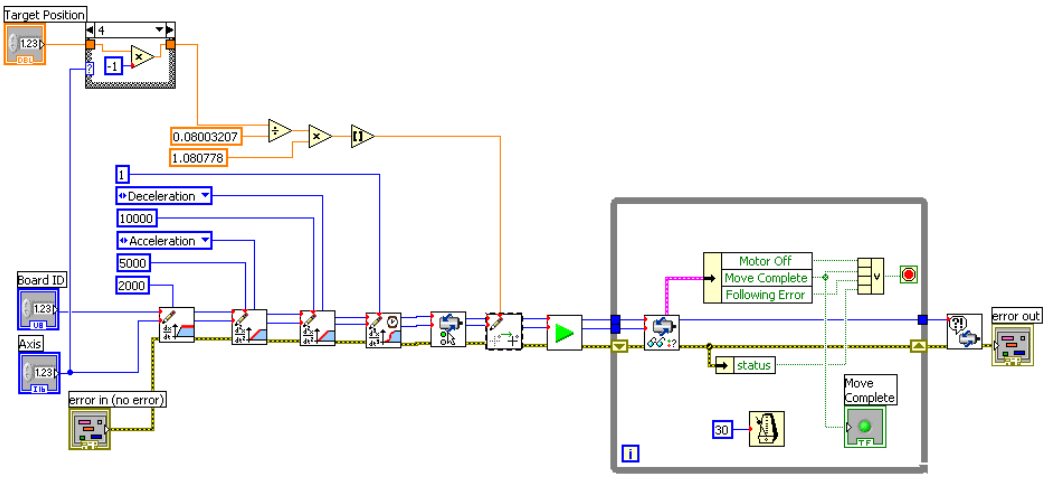


Figure 2.14: NI Motion Control toolbox used to move stepper motors.

on the serial port and stored on a register. It is then transferred to a variable in memory and converted into a pose variable. The pose variable is passed to a move function, which sends the robot to the position described by the pose variable. The register is finally cleared and the program ends.

Controlling the robot in this way requires well-timed signalling between the Labview software running on the PC and the robot program running on the AX controller. However this is not a problem as Labview provides timers and sequence structures that allow one to accurately control when a flag is set or variable is sent to the AX controller etc.

Chapter 3

Stereo Vision Programming

Parallel research on the dairy parlour robot project focussed on automated teat identification using thermal imaging to find the teats and a stereo pair of cameras to measure their positions [19]. The measurements produced showed sufficient accuracy for the application and it was decided to integrate the stereo vision system into the control system of the robot to provide target positions for the placement of the milking cups. In order to understand the challenges surrounding the acquisition of data accurate enough for this task, an explanation of the geometry and parameters associated with the cameras is needed.

3.1 Theory

3.1.1 Single View Geometry

The simplest model of a camera is known as the pinhole camera. This is represented in Figure 3.1. It consists of two screens arranged in parallel, one with a small hole in the centre. The hole is theoretically a point. The object to be imaged is placed in front of this. Rays of light, either emitted from or reflected off the object, pass through the hole and the image of the object projected on to the back screen is inverted in the process.

To introduce some terminology, the back screen is called the image plane and the

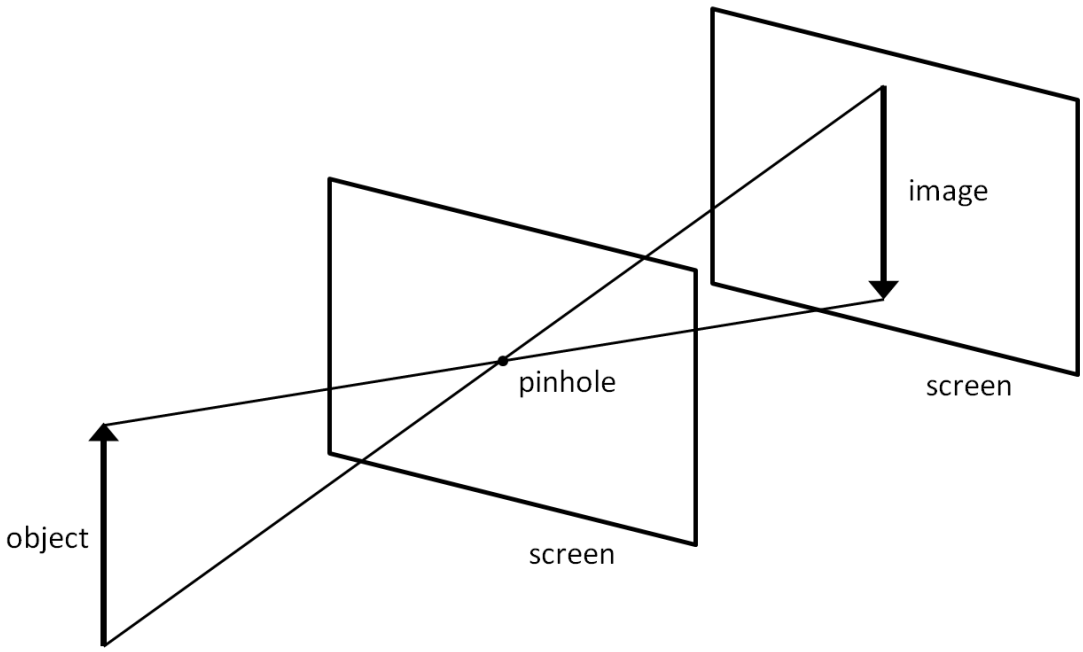


Figure 3.1: Theoretical model of a pinhole camera.

position of the hole is called the camera centre. The point on the image plane orthogonally opposite the camera centre is called the principal point and the line joining the two is the principal axis. The distance between the two screens (i.e. between the camera centre and the principal point) is the focal length.

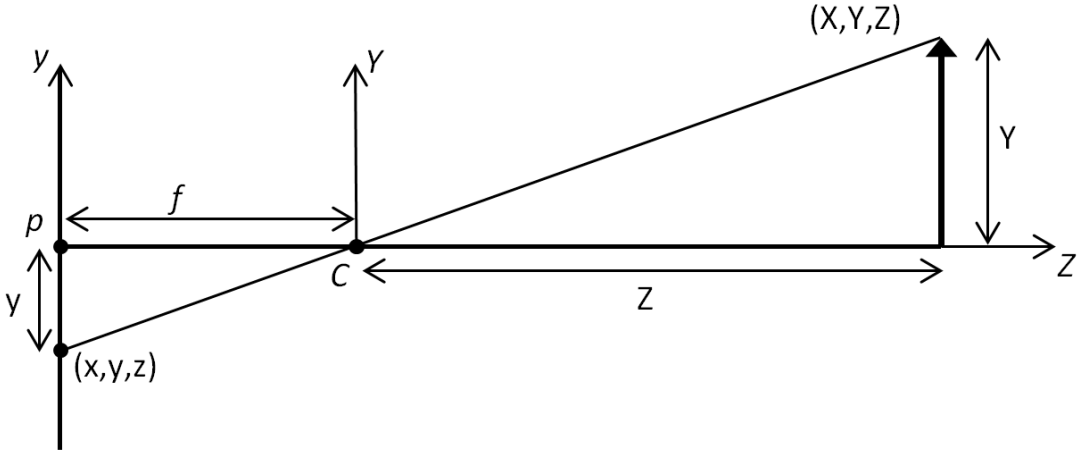


Figure 3.2: Side view of pinhole camera model.

Figure 3.2 is a representation of Figure 3.1 seen from the side. Coordinate frames have been added, indicated by italicised letters. The principal point is denoted p, the

focal length is f and the camera centre is C. The 3D point (X, Y, Z) is projected through the camera centre and its image is (x, y, z). From this diagram, the height of the image, y, is given by:

$$\frac{y}{f} = \frac{Y}{Z} \tag{3.1}$$

$$y = \frac{fY}{Z} \tag{3.2}$$

If Figure 3.2 is rotated 90° about the principal axis, the horizontal coordinate (x-axis) of the image point can be found the same way to be:

$$x = \frac{fX}{Z} \tag{3.3}$$

Equations (3.2) and (3.3) represent the 2D image of a 3D point. They can be combined in matrix form as [43]:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{3.4}$$

Or:

$$x = PX \tag{3.5}$$

where $P = [K|0]$and K is a 3×3 matrix known as the camera calibration matrix. These equations assume that the origin of the image plane coordinate system is at the principal point but often it is not. Equation (3.4) is adjusted for this by adding offsets in the image x- and y-directions, p_x and p_y respectively. There is also an assumption that pixels are square, i.e. that the axes on the image plane have the same scale. Since this

is not necessarily the case, the scaling factors m_x and m_y are incorporated. Further, it is possible that the axes are not perpendicular so a skew factor s is included. K now becomes:

$$K = \begin{bmatrix} \alpha_x & s & 0 \\ 0 & \alpha_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

where $\alpha_x = m_x f$ and $\alpha_y = m_y f$ [44].

The transformation matrix from the camera coordinate frame to the world coordinate frame is incorporated into the matrix P such that:

$$P = K[R| - R\tilde{C}] \tag{3.7}$$

where R is the 3×3 matrix of rotations and $\tilde{C}$ is the camera centre given in world coordinates.

The derivation of the intrinsic parameters, K, is discussed in greater detail in [45] and the extrinsic parameters, R and C, in [46].

3.1.2 Two View Geometry

The previous section discussed how to model a camera such that a point in 3D space can be transformed onto a 2D image plane. The stereo vision problem breaks down into two issues. Firstly, given a point x on the image plane of one camera, the corresponding point on the image plane of the other camera, x', must be found. This is known as the correspondence problem. Secondly, given x and x', the corresponding 3D point X must be found in world coordinates. This is the reconstruction problem [47].

Figure 3.3 represents two pinhole camera models. Note that the camera centres C and C' are now behind the image planes. Previously they were in front of them (Figure 3.2). This is simply an alternate representation of the camera model: if the camera centre C is

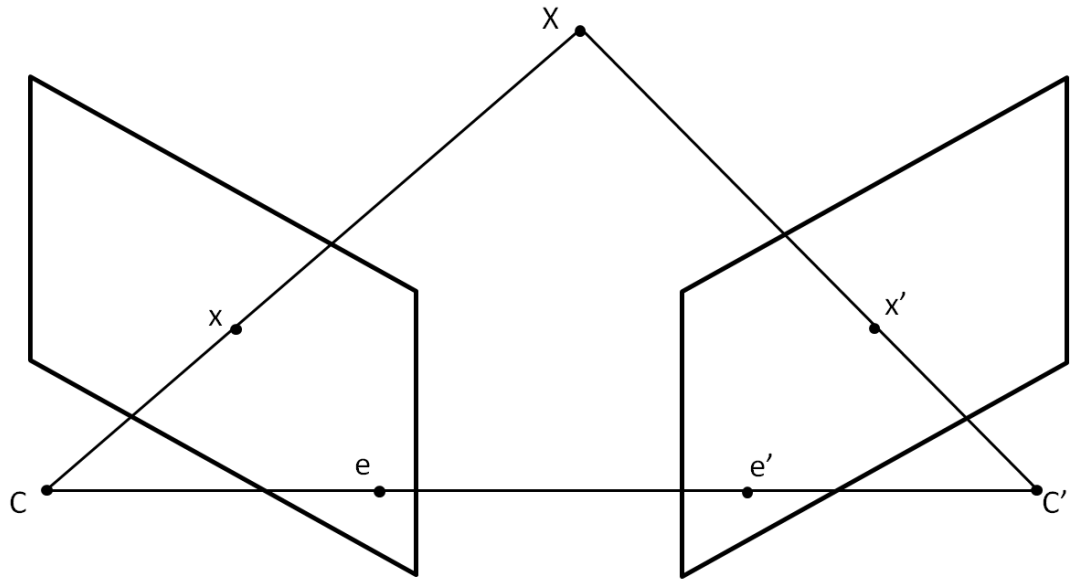


Figure 3.3: Stereo camera views.

moved to the other side of the image plane at a distance of f (focal length), the geometry will be the same. The previous version is used in [38] and this version is used in [37].

The 3D point X is projected onto the image planes through the camera centres C and C' to create images x and x'. The image of the right camera centre on the left camera plane is the epipole, e, and the opposite is e'. The line connecting the two camera centres is the baseline. The plane containing the baseline and the point observed is the epipolar plane. The epipolar line is the intersection of the epipolar plane and the image plane.

The solution to the correspondence problem is given in Equation (3.8):

$$x'^{T} F x = 0 \tag{3.8}$$

It should be noted that since e' indicates the point in the right hand image corresponding to the point e in the left hand image, the transpose is here indicated with a superscript 'T'. The matrix F in Equation (3.8) is called the fundamental matrix and is given as:

$$F = [e']_{\times} H_{\pi} \tag{3.9}$$

where $[e']_\times$ is the skew-symmetric matrix of the epipolar point e'. H_π is a 2D homography mapping x to x'. The solution to the correspondence problem is derived in [48].

The coordinates of the 3D point are reconstructed using the image coordinates of the points x and x' and the projection matrices P and P'. Each camera has an equation relating the image coordinates to the 3D coordinates through the projection matrix (Equation (3.5)). These equations are combined to give:

$$AX = 0 \tag{3.10}$$

where A is given as:

$$A = \begin{bmatrix} x_x p_3 - p_1 \\ x_y p_3 - p_2 \\ x'_x p'_3 - p'_1 \\ x'_y p'_3 - p'_2 \end{bmatrix} \tag{3.11}$$

Here, x_x is the x-component of the point x etc. and p_i is the i^{th} row of the projection matrix, P. This method is derived in [49] and is a very simplistic way of triangulating a point since it assumes that rays projected from x and x' through their respective camera centres coincide at X. Practically however, the presence of noise on the cameras will mean the rays found will not necessarily cross. The triangulation routine used in this application from the Bouguet calibration toolbox [39] finds the best estimate in the presence of noise. This toolbox is discussed below.

3.1.3 Calibration

The calibration of a single camera is the process of finding the values of the projection matrix, P, as defined in Equation (3.5). This equation maps a point X_i in 3D space to

a point x_i on the image plane. It can be expanded as:

$$\begin{bmatrix} su_i \\ sv_i \\ s \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & p_{14} \\ p_{21} & p_{22} & p_{23} & p_{24} \\ p_{31} & p_{32} & p_{33} & p_{34} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \\ 1 \end{bmatrix} \tag{3.12}$$

$$\begin{bmatrix} su_i \\ sv_i \\ s \end{bmatrix} = \begin{bmatrix} p_{11}x_i + p_{12}y_i + p_{13}z_i + p_{14} \\ p_{21}x_i + p_{22}y_i + p_{23}z_i + p_{24} \\ p_{31}x_i + p_{32}y_i + p_{33}z_i + p_{34} \end{bmatrix} \tag{3.13}$$

where $x_i(u_i, v_i)$ is the point on the image plane and $X_i(x_i, y_i, z_i)$ is the 3D point. From this then we get:

$$u_i(p_{31}x_i + p_{32}y_i + p_{33}z_i + p_{34}) = p_{11}x_i + p_{12}y_i + p_{13}z_i + p_{14} \tag{3.14}$$

and:

$$v_i(p_{31}x_i + p_{32}y_i + p_{33}z_i + p_{34}) = p_{21}x_i + p_{22}y_i + p_{23}z_i + p_{24} \tag{3.15}$$

Given six pairs of corresponding points x_i and X_i, we then have twelve simultaneous linear equations. From these, the twelve elements of the projection matrix, P, are found. This again assumes errorless correspondences between real world points and image points. As mentioned in the previous section, due to the presence of noise, this may not be the case. It is possible to over-determine the solution by using more than six pairs of points. In this case, the more points that are used, the more accurate the resulting projection matrix, P, will be [50].

The fundamental matrix, F, was defined in the previous section (Equation (3.8)). This matrix represents the epipolar geometry - the spatial relations between the two

cameras. Equation (3.8) can be expanded as:

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = 0 \tag{3.16}$$

$$\begin{bmatrix} u' & v' & 1 \end{bmatrix} \begin{bmatrix} uf_{11} + vf_{12} + f_{13} \\ uf_{21} + vf_{22} + f_{23} \\ uf_{31} + vf_{32} + f_{33} \end{bmatrix} = 0 \tag{3.17}$$

$$u'uf_{11} + u'vf_{12} + u'f_{13} + v'uf_{21} + v'vf_{22} + v'f_{23} + uf_{31} + vf_{32} + f_{33} = 0 \tag{3.18}$$

Where $x(u, v)$ and $x'(u', v')$ are corresponding points on each camera. As with the calibration of the individual cameras (i.e. finding the projection matrix, P), point correspondences will not be exact due to noise and an over-determined system of equations is used. This means using as many points as possible to reduce the error in F caused by correspondence errors [51].

3.2 Development

The development of the vision system is based on previous work [49][19]. In [19], a thermal camera was used to identify the teats of a heated dummy rig. The stereo pair of cameras was then used to calculate their positions. The problem of automatically identifying the teats was beyond the remit of the current research project. In this project the teats are manually selected from the camera feeds to provide point correspondences for each teat.

3.2.1 Design of Camera Rig

The cameras used in this application were the Prosilica EC1350 firewire cameras used in [19]. These have a resolution of 1360×1024 pixels. These were mounted on a beam to have parallel principal axes and a baseline length of 100mm. This was based on the setup used in [19]. The beam was mounted via a tapped hole in its centre to a camera ball mount. That was in turn fixed to the floor in a position that allowed the cameras a field of view wide enough to cover the full range of likely teat positions but small enough to provide large enough views of the teats.

As development continued and testing was performed to ascertain the accuracy of the stereo vision system, it was decided that a wider baseline was needed along with the ability to rotate the principal axes of the cameras relative to one another. An older stereo vision rig used in an earlier phase of research [49] was then repurposed. This original vision system was found to perform poorly and the rig had been abandoned. It included a mechanism for adjusting the baseline of the stereo pair and a hinged mount to allow one of the cameras to be rotated relative to the other. This was then modified to accept the two Prosilica cameras. The baseline could now be extended and one camera rotated inwards so that the two cameras had a similar field of view. This rig already had a 1/4" UNC tapped hole drilled into the base to allow it to mount directly to the ball mount. This size bolt is standard in photographic applications. The final stereo vision setup is shown in Figure 3.4 mounted on the floor.

During development the possibility of improved results by mounting the cameras on the end-effector was investigated. This was prompted by the thought that if the cow was in an unusual position in the stall, the cameras could be moved to give a clear set of images. It would allow more flexibility in the presence of obstructions such as the other equipment in the dairy parlour. The cameras would also be able to get closer to the teats, in theory, allowing larger images and more accurate results.

An assembly was designed in ProEngineer and fabricated in the workshop of the School of Mechanical Engineering that consisted of a mounting bracket and an adjustable beam.

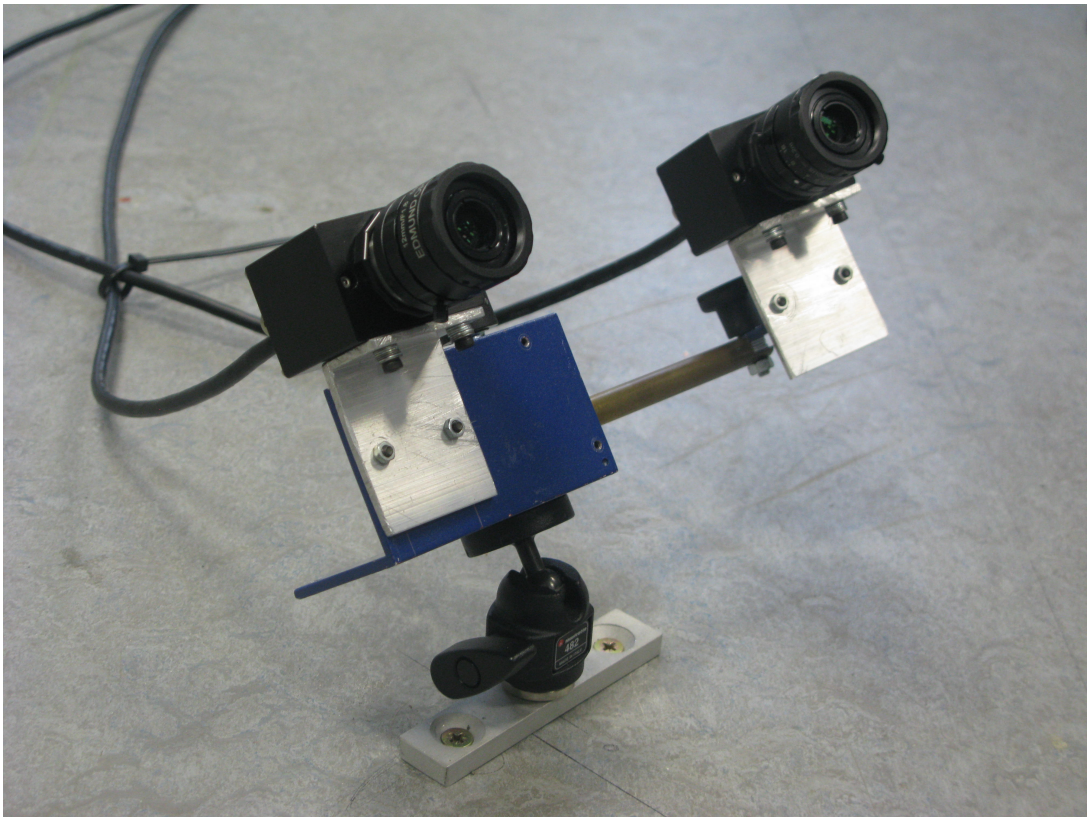


Figure 3.4: Stereo vision system mounted to floor.

The mounting bracket was bolted to the side of the end-effector using existing holes. The adjustable beam, to which the stereo rig was bolted, allowed the cameras to be tilted about an axis parallel to the y-axis of the robot coordinate frame. This setup is shown in Figure 3.5.

3.2.2 Calibration Procedure

To use the pair of cameras to measure points in 3D space they must first be calibrated to obtain both the intrinsic parameters for each camera as well as the extrinsic parameters. These are referred to as the projection matrix, P, and the fundamental matrix, F, respectively in Section 3.1. It was not necessary to write software to perform the calculations for this as there are several camera calibration toolboxes available online such as OpenCV [52] and GML C++ Camera Calibration Toolbox [53]. The one used in previous research on this project [19] was the Caltech Camera Calibration Toolbox for Matlab, developed

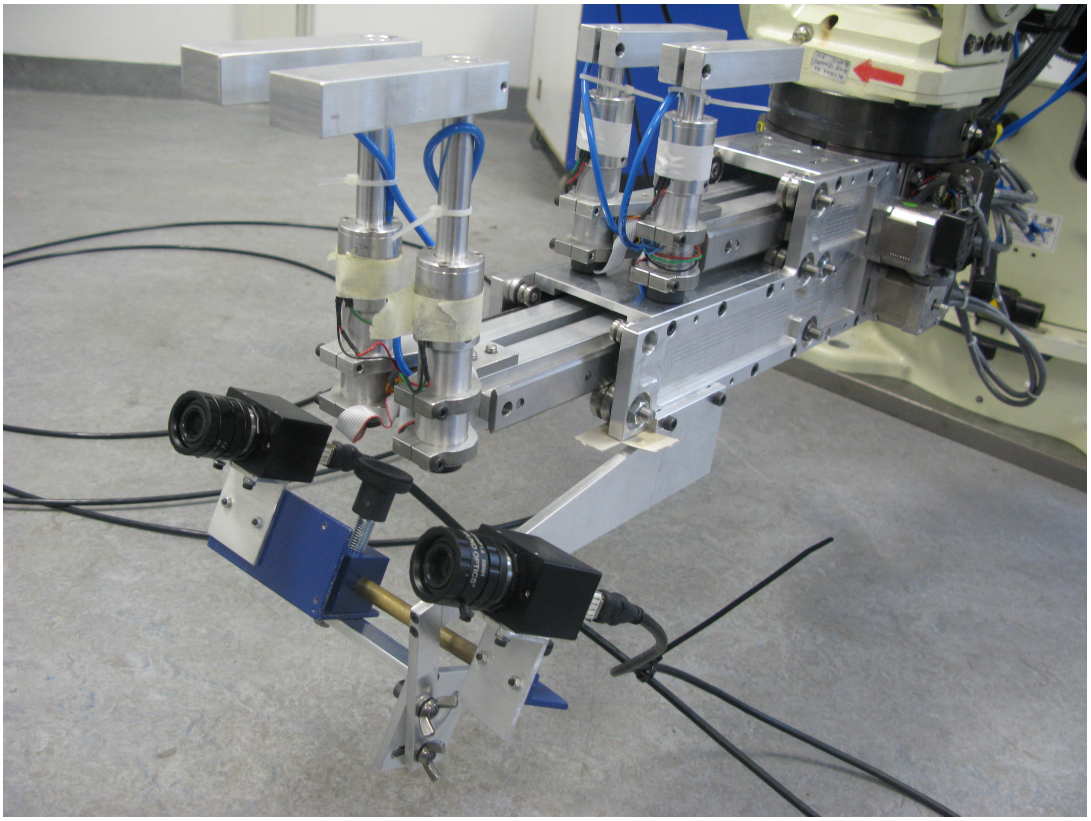


Figure 3.5: Stereo vision system mounted to end-effector.

by Jean-Yves Bouguet. In the previous research this appeared to give good results. Also, it being implemented in Matlab made it convenient to use in conjunction with Labview via the MathScript node. This is a function block that allows scripts written in Matlab to be run in the Labview environment with data being passed in from the system and data returned.

The Bouguet toolbox uses a set of parameters that define a camera model similar to that used by Heikkilä and Silvén [40]. This is similar to the model developed in Section 3.1, with the addition of a distortion model. This accounts for errors introduced by imperfections in the lens i.e. radial and tangential. This model of distortion is known as the plumb line model [41].

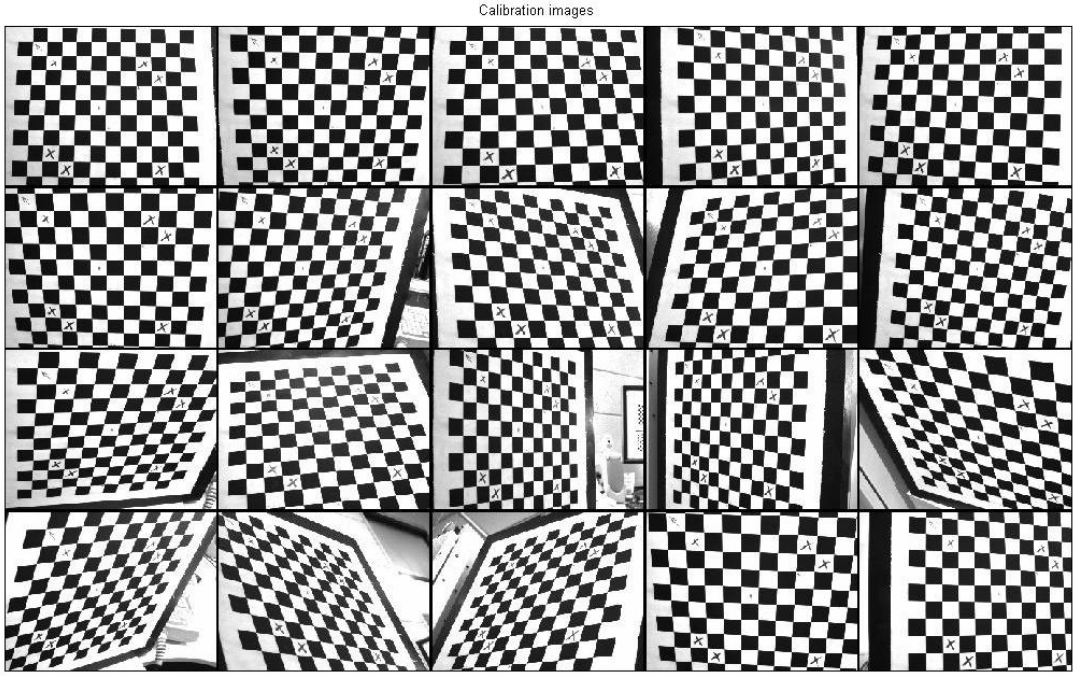
An essential part of finding the intrinsic and extrinsic parameters is to find a large enough set of point correspondences. For the intrinsic matrices, these are correspondences between image points and points in 3D space and for the extrinsic parameters they

are correspondences between the image coordinates for each camera. The image pixel coordinates are calculated by Matlab. The corresponding points in 3D space must be known and to do this a printed checkerboard pattern of known dimensions is used. This allows the relative positions of the points selected in the image to be known in a 3D system of coordinates. The fact the checkerboard is planar presents a limitation on the set of points so to counteract this, and to gather more points overall, multiple images are taken with the board at various orientations.

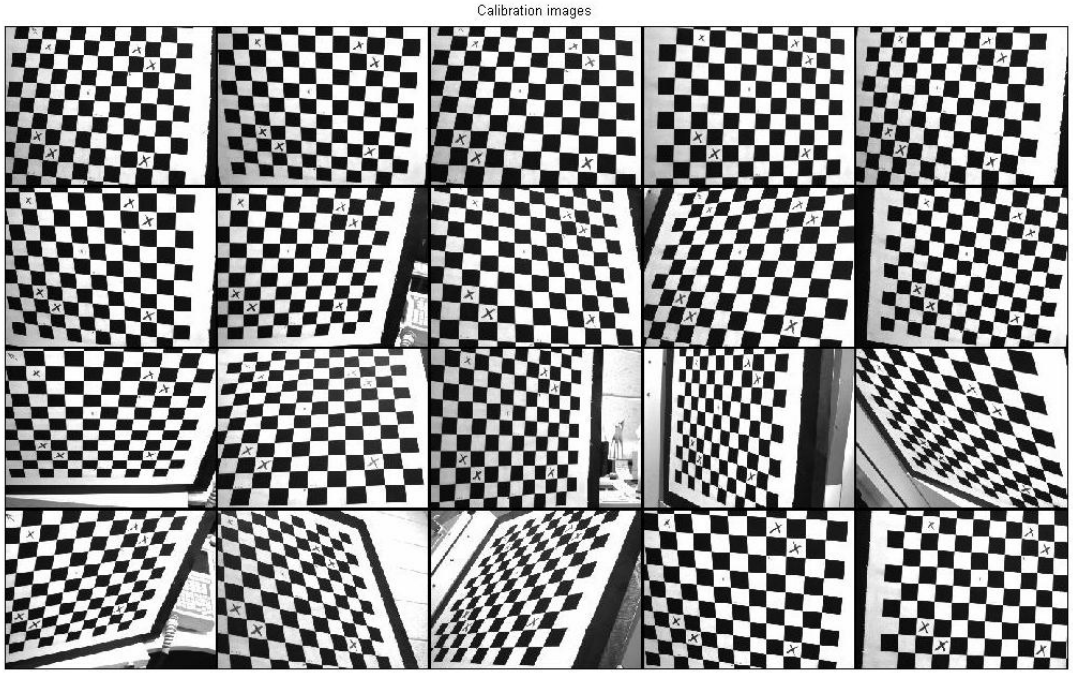
The complete procedure for calibrating the cameras is described at [39], however here we will discuss the main aspects. The Labview virtual instrument 'Grab and Snap - 002.vi', the development of which is discussed in Section 3.2.3, is used to acquire a set image pairs taken by the cameras. In all, 20 pairs of images are taken. The cameras are each positioned and focussed to give optimal views of the teats. The checkerboard is kept at a distance about equal to the working distance of the cameras. These images are loaded into memory by the calibration toolbox.

Figure 3.6 shows an example of an image set taken for calibration. The upper set of images, Figure 3.6a, is from the left camera, the lower, Figure 3.6b, from the right. Not all squares on the checkerboard are visible since only an 8×8 set of squares was used. This provided 81 corners to be identified. In the Bouguet calibration routine, the user selects the four corners of the 8×8 set (any number of squares can be used). The rest of the corners are then extrapolated using straight lines with the user given an option of providing an initial estimate for distortion. Once this has been performed for all images, the toolbox uses a corner detector to find the corners. The purpose of this is to calculate the distortion of the lens. If the extrapolated corners are far from the detected ones, distortion is high. Thus by correlating the pixel coordinates of each corner with the coordinates in 3D space, the Bouguet routine can calculate the parameters of the camera model and hence the calibration matrix for each camera. It is also able, given the intrinsic parameters for each camera, to find the extrinsic parameters between them.

Figure 3.7 shows the reprojection error for one of the images. This is the left image



(a) Left camera



(b) Right camera

Figure 3.6: Calibration image pairs.

from the eighth pair. The reprojection error is the error between the extrapolated points and those found by corner detection, indicated by size and direction of the arrows. A good initial estimate of distortion error removes a good portion of the systematic error but some remains, as indicated by adjacent points having similar errors. The reprojection errors for all points in all images taken by one camera are shown in Figure 3.8. The error for each point is shown as a cross, with the magnitude of the error on each axis. Points of the same colour are on the same image. Figure 3.8 shows no particular image causing larger errors than the rest and overall the errors are less than half a pixel in each direction. This data is presented again in Figure 3.9 where it is divided into two histograms showing the errors on each axis. This shows the reprojection errors to have distributions approaching normal with a mean of less than 0.1 pixels.

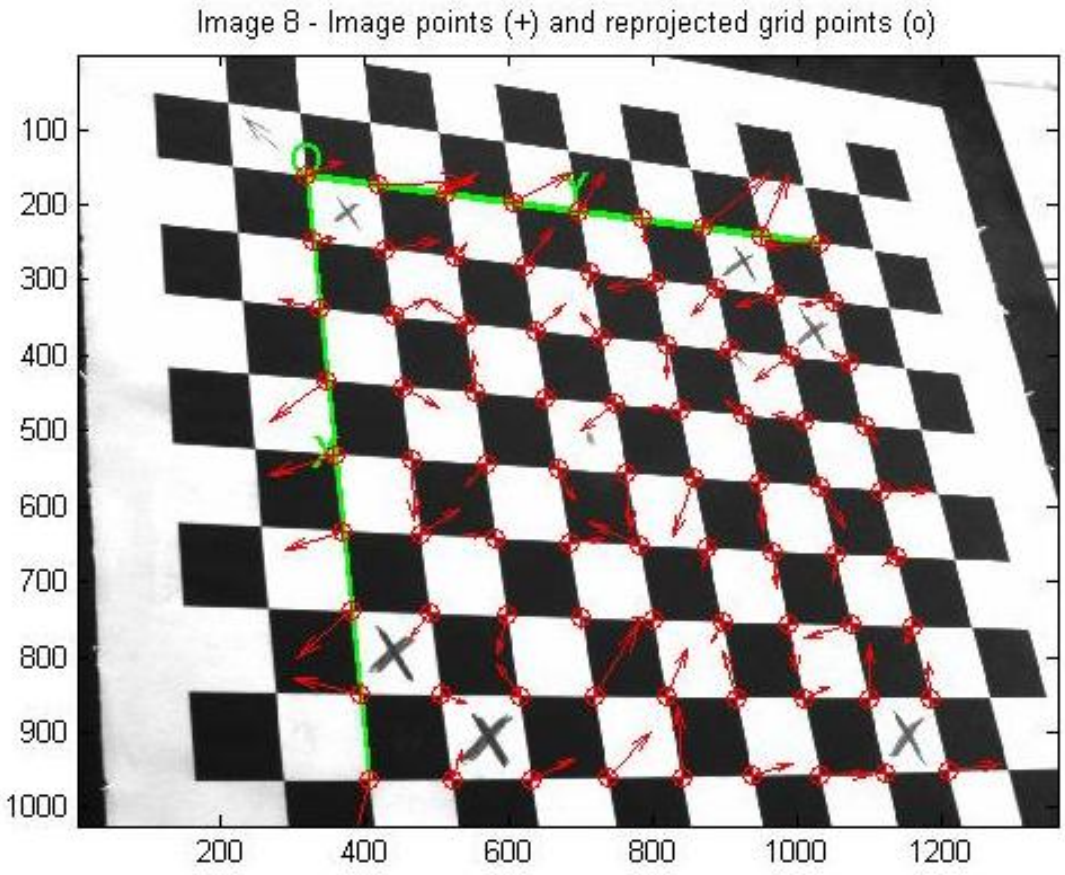


Figure 3.7: Reprojection errors on one image.

The Bouguet toolbox also gives the option of visualising the radial, tangential and

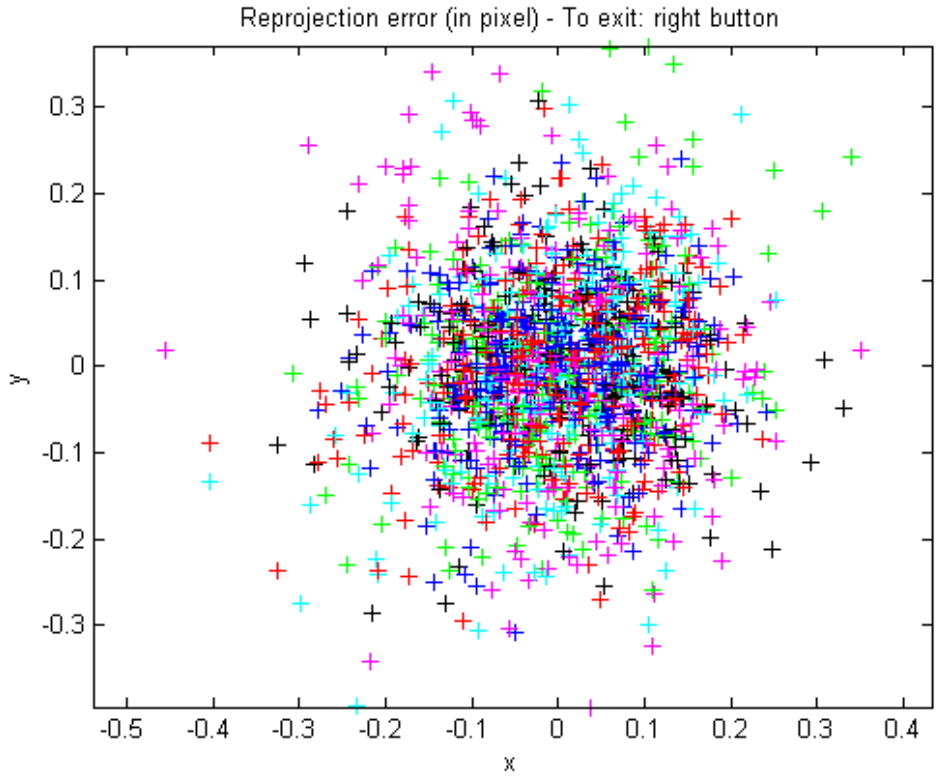
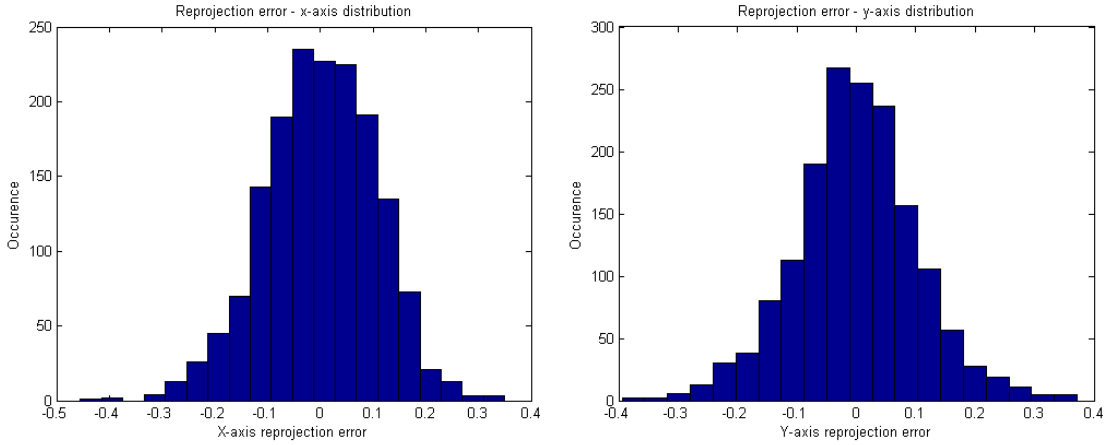


Figure 3.8: Reprojection errors for all points on all images taken by one camera. Units are pixels.



(a) X-axis distribution

(b) Y-axis distribution

Figure 3.9: Reprojection errors for all points across all images approaching normal distributions in both x- and y-directions. Units are pixels.

combined distortion. The radial and tangential distortions are shown in Figure 3.10. The error due to radial distortion is up to 14 pixels at the top left of the images, Figure 3.10a. Due to tangential distortion, the error reaches 0.9 pixels - a far smaller error, Figure 3.10b. These errors may seem substantial but they are being corrected by the distortion model.

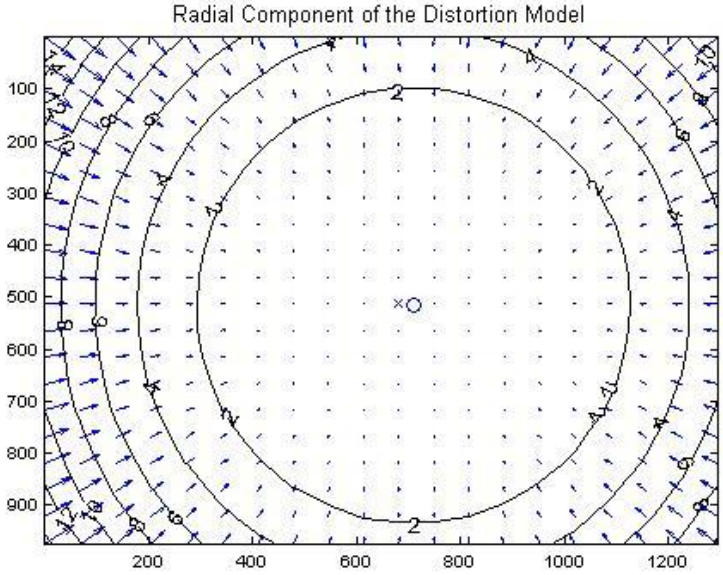
The results of the calibration corresponding to the previous figures are shown in Table 3.1. This shows the intrinsic parameters for both cameras. The Bouguet toolbox also returns a measure of the degree of uncertainty associated with each parameter. The focal length is given as a 2×1 vector, the elements of which are the values αx and αy incorporating the aspect ration of the pixels. The skew coefficient, a measure of angles between the axes of the pixels, is assumed to be 0 since it is uncommon for a camera to have non-square pixels, according to Bouguet.

Table 3.1: Typical intrinsic calibration parameter values.

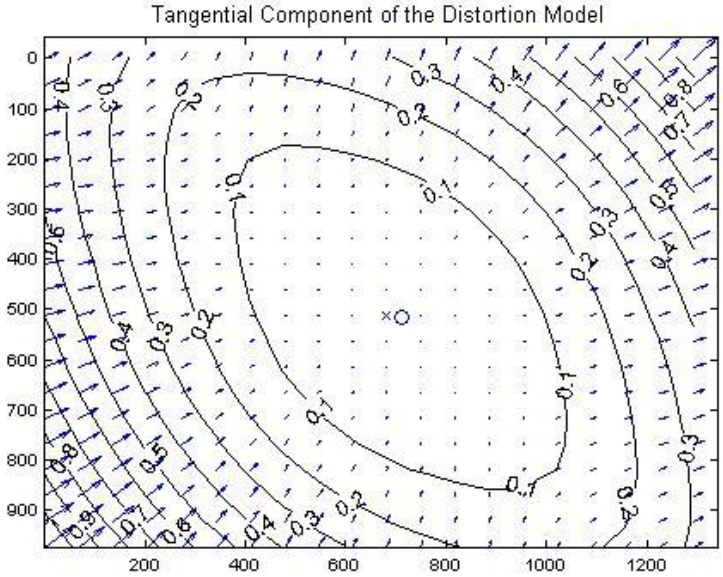
	Left Camera		Right Camera	
		Uncertainty		Uncertainty
Focal Length, fc (mm)	2588.847 2593.655	1.188 1.264	2589.698 2594.475	1.229 1.282
Principal Point, cc (mm)	708.881 515.996	2.126 2.011	711.533 520.887	2.161 2.046
Skew Coefficient, alpha_c	0	0	0	0
Distortion Coefficients, kc	-0.189 0.264 -0.001 0.001 0	0.005 0.062 0.0002 0.0002 0	-0.183 0.237 0.0002 -9E-06 0	0.004 0.041 0.0002 0.0002 0

The extrinsic parameters were returned as a set of translations along each axis, T, such that T = [-167.2018; -0.0786; -8.5913]. The rotations are given as a set of rotations about each axis, om = [0.0428; 0.1886; 0.0446].

It was found during development that the quality of the images used in the calibration had a non-trivial effect on the eventual errors of the system. Every time the calibration was performed different results would be found, some better than others. There was thus



(a) Radial



(b) Tangential

Figure 3.10: Distortion errors across the image plane. X- and y-axes represent pixel values.

a learning curve involved in using the calibration toolbox and consistent results were not always achieved.

3.2.3 Vision System Software Design

The ultimate goal of the software design of the vision system was to be able to have a live camera feed from the two cameras displayed simultaneously so that a set of corresponding points in each image can be selected using the mouse and the 3D world coordinates of those points returned. The first step in the development was to create a Labview virtual instrument (VI) capable of acquiring a video feed from a camera and saving an image to a file on demand. The block diagram for this first VI is shown in Figure 3.11.

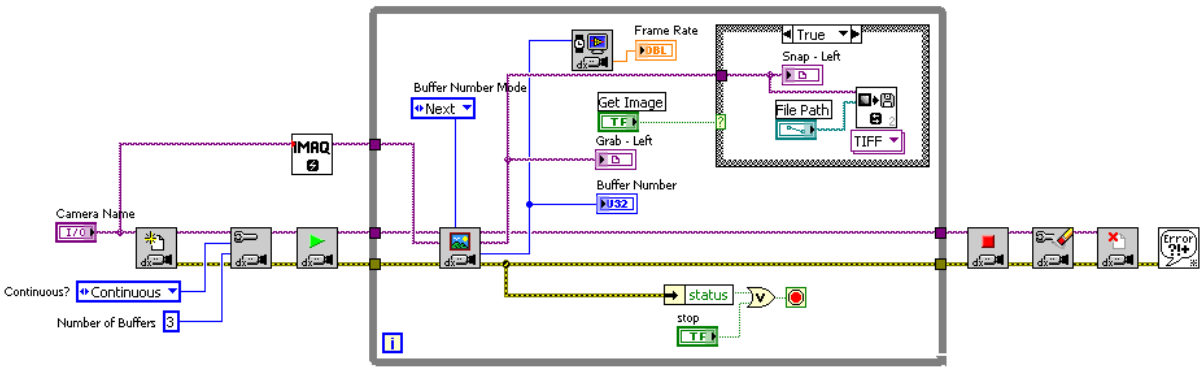


Figure 3.11: Labview block diagram of image capture software.

This initialises contact with the camera, configures the communication and begins acquisition of images. Inside the while loop, a frame is taken from the video feed and saved to a specified file path when the SNAP button is clicked. When the STOP button is pressed, the while loop exits and reverses the initialisation procedure performed before the while loop. The front panel of this VI is shown in Figure 3.12, showing the video feed in the upper panel and the saved image in the lower panel.

A new VI was created titled 'Grab and Snap - 002.vi', that effectively doubled up the block diagram shown in Figure 3.11 with a common while loop. This allowed images from both cameras to be captured simultaneously. Further functionality was included

Figure 3.12: Labview front panel of image capture software.

that allowed the pixel coordinates of mouse clicks to be output by the VI. This was done by including an event structure for each camera. The event structure in Labview waits for a certain event to occur, such as key strokes or actions of the mouse, and then executes a subdiagram. The event structure and relevant subdiagram is shown in Figure 3.13. In this case, the event structure waits for the mouse to click on a point within the panel of the captured image. The subdiagram in this event provides a reference to the image panel and extracts the last mouse position property. The pixel coordinates are stored in an array that is initialised outside the loop.

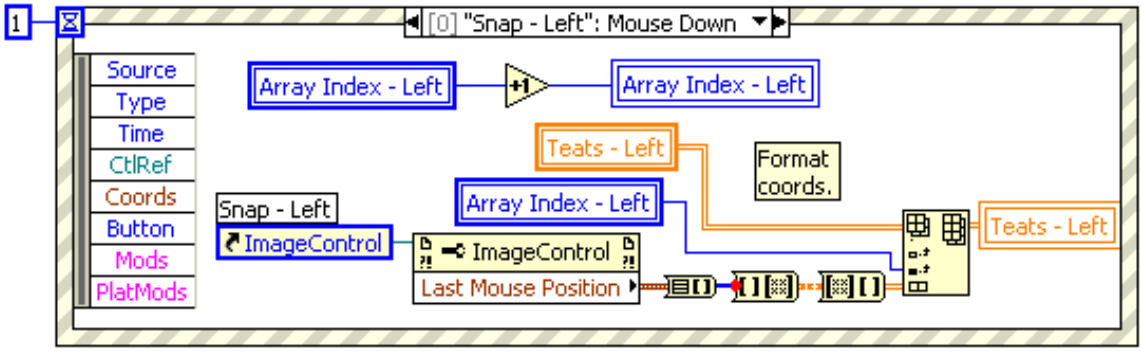


Figure 3.13: Event structure - store pixel coordinates when the mouse clicks on the image.

The next step in the development of the vision system was to use the pixel values to determine the 3D coordinates of the points being measured. The Bouguet toolbox provides a Matlab script called stereo_triangulation.m. This accepts as inputs the pixel coordinates of the selected points, denoted in Section 3.1 as $x(u, v)$ and $x'(u', v')$. It also requires the intrinsic parameters for each camera and extrinsic parameters. These were found using the calibration routine described in Section 3.2.2. The new triangulation VI was composed of a Mathscript node with inputs and outputs. The contents of stereo_triangulation.m were copied into the Mathscript node and modified so that the intrinsic and extrinsic parameters were defined in the script as constants, since these would not be changing during normal operation of the system. (If any of these parameters were changed, such as by refocusing the cameras or repositioning them relative to one another, the calibration routine would need to be repeated on the stereo pair). A problem arose at

this point that prevented the VI from running without errors. It turned out to be caused by a difference in the way Labview and Matlab handle one-dimensional arrays. Labview assumes that a 1D array is a column whereas Matlab assumes they are rows. This can be demonstrated by passing a 1D array, IN, from Labview into a Mathscript node. The only contents of the node are 'OUT = IN'. The value OUT is then output. This resulting variable OUT is transposed to a row, despite there being no transposition operation performed. Once this problem was identified and the triangulation script corrected, the VI ran without errors. The two sets of pixel values produced a set of 3D coordinates that appeared to show realistic values. The 3D coordinates are defined in a reference frame centred on the camera centre. The accuracy of the coordinates produced could not be tested until the transformation from the stereo cameras' coordinate frame to the world frame was found. The block diagram of the triangulation VI is shown in Figure 3.14.

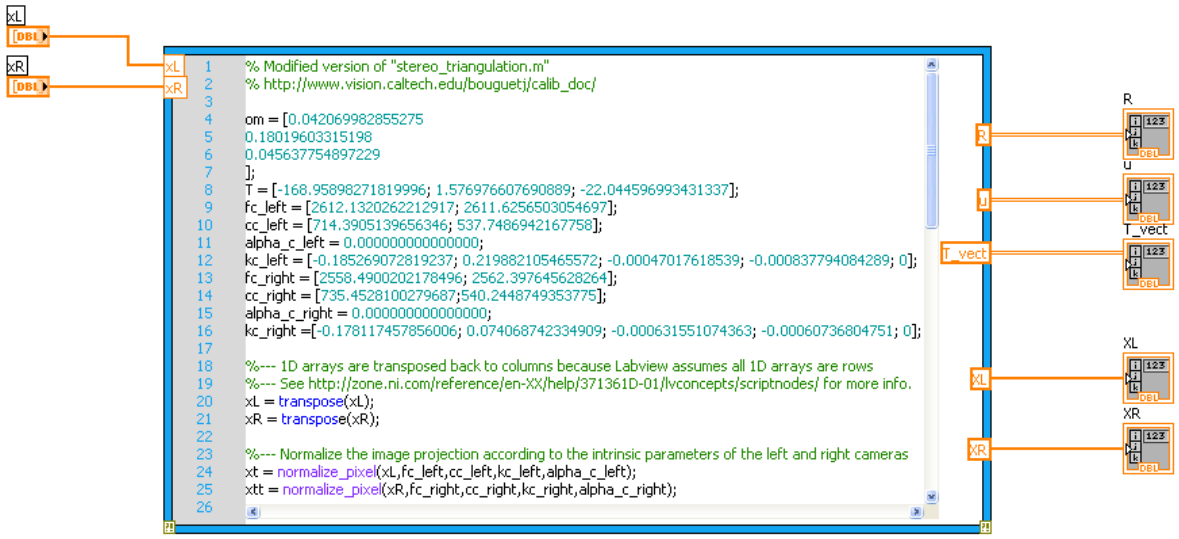


Figure 3.14: Triangulation software in Matlab code included in a Labview VI using a Mathscript node.

3.2.4 Spatial Transformations

In order to express the 3D coordinates found by the vision system in the world frame a transformation matrix was needed. The world coordinate frame was chosen to coincide

with the coordinate frame of the 6-axis robot. Target positions for the robot are given to the robot controller in this coordinate frame. It makes sense then to use this as the frame in which teat coordinates are specified to the actuation system. The vision system coordinate frame is shown relative to the robot frame in Figure 3.15.

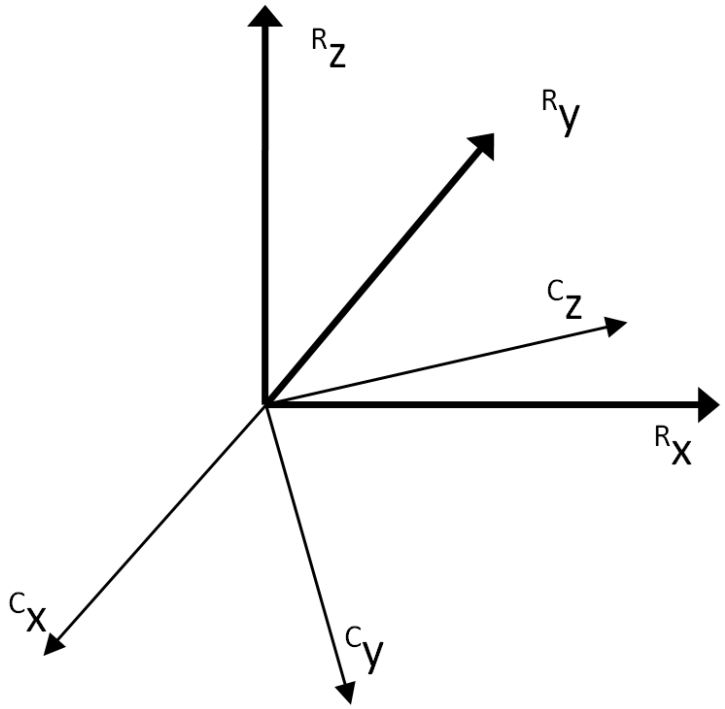


Figure 3.15: Robot and vision system coordinate frames shown with coincident origins. In reality there is an offset between the origins.

Two methods of creating such a transformation matrix were identified. In the first, the robot was moved to a set of known positions and a point on the end-effector was measured using the vision system. This provided a cloud of points with their positions given in both robot and camera frames. A least-squares operation in Matlab provided the transformation between the two.

In the second method, the following procedure was used:

1. Select a point on the end-effector that is easily identified in camera images. This point should be a constant offset from the centre of the robot toolplate (this is the endpoint of the robot that is moved to target positions). The known position of this point in the robot frame is called ${}^{R}p$.

2. The selected point is moved into the field of view of the cameras and its position measured in the camera frame. This observed point will be referred to as ${}^{C}p$, in the camera frame.

3. The end-effector is moved 100mm along the x-axis of the robot frame and the resulting position is measured with the cameras, ${}^{C}x$.

4. The change in coordinates, ${}^{C}x - {}^{C}p$, is divided by its magnitude to result in a unit vector, X, along the robot x-axis given in camera coordinates.

5. Steps 3 and 4 are repeated for the y- and z-axes to result in the corresponding unit vectors, Y and Z.

These three unit vectors define the ${}^{C}Rot_{R}$ matrix. This is a 3×3 matrix that determines the rotation of the robot frame relative to the camera frame. This is shown in Equations (3.19) - (3.22).

$$X = \frac{{}^{C}x - {}^{C}p}{|{}^{C}x - {}^{C}p|} \tag{3.19}$$

$$Y = \frac{{}^{C}y - {}^{C}p}{|{}^{C}y - {}^{C}p|} \tag{3.20}$$

$$Z = \frac{{}^{C}z - {}^{C}p}{|{}^{C}z - {}^{C}p|} \tag{3.21}$$

$${}^{C}Rot_{R} = \begin{bmatrix} X & Y & Z \end{bmatrix} \tag{3.22}$$

Here, ${}^{C}x$ is the observed point measured in camera coordinates having been moved 100mm along the robot x-axis. ${}^{C}y$ has been moved along the y-axis and ${}^{C}z$ along the z-axis. The vector from the known point ${}^{R}p$ to the robot frame origin, ${}^{R}O$, is rotated using the rotation matrix ${}^{C}Rot_{R}$ to align with the camera frame. The position of the

origin of the robot frame, ${}^{C}R$, is then the sum of the shift from the camera origin to the observed point and the shift from the observed point to the robot origin.

$$ {}^{C}R = {}^{C}\,p + {}^{C}\,Rot_R * ({}^{R}O - {}^{R}\,p) \tag{3.23} $$

This is combined with the rotation matrix, ${}^{C}Rot_R$, (and a row [0 0 0 1]) to give the 4×4 transformation matrix ${}^{C}T_R$, which describes the robot coordinate frame in camera coordinates. This is inverted to give ${}^{R}T_C$. This is the transformation matrix that will convert points in the camera coordinate frame to points in the robot frame. This is equivalent to:

$$ {}^{R}T_C = \begin{bmatrix} X & Y & Z & {}^{C}O \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \tag{3.24} $$

$$ = \begin{bmatrix} X_x & Y_x & Z_x & {}^{C}O_x \\ X_y & Y_y & Z_y & {}^{C}O_y \\ X_z & Y_z & Z_z & {}^{C}O_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \tag{3.25} $$

A set of tests was performed to evaluate each method. The teats on the experimental rig were moved to a set of positions and measured using the Microscribe and the vision system. The vision system measurements were transformed using matrices found using the two methods. The results are summarised in Table 3.2 and show that the techniques have similar mean and standard deviations. However, the cloud-based techniques exhibits a far larger mean error on the z-axis and a larger standard deviation on the x-axis. It was thus decided to use the method based on four points.

When the cameras are mounted on the end-effector the position of the camera coordinate frame changes as the robot moves. The transformation matrix mapping points from the camera coordinate frame to the robot coordinate frame, ${}^{R}T_C$, was recalculated every time the robot stopped and a measurement was taken. The camera coordinate

Table 3.2: Errors of measurements of a set of 25 points found using two transformation matrices.

	Using 4 selected points:			Using point cloud:		
	x	y	z	x	y	z
Mean, (mm)	-3.823	-1.131	-0.226	-3.285	1.406	-2.067
Standard Deviation, σ (mm)	7.845	1.893	8.475	10.604	1.578	8.199

frame has a constant offset from the centre of the toolplate. The centre of the toolplate is moving relative to the robot coordinate frame, but its position is always known. Thus if the translation from the robot frame to the toolplate is known and we find the transformation from the toolplate to the camera coordinate frame, the overall transformation from the robot to the cameras can be given by:

$$^{R}T_{C} = ^{R}T_{f} * ^{f}T_{C} \tag{3.26}$$

where,

$$^{R}T_{f} = \begin{bmatrix} 1 & 0 & 0 & ^{R}x \\ 0 & 1 & 0 & ^{R}y \\ 0 & 0 & 1 & ^{R}z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.27}$$

such that $T(^{R}x, ^{R}y, ^{R}z)$ is the centre of the toolplate in robot coordinates. The matrix $^{f}T_{C}$ is found using the following procedure.

1. Select a point fixed in space relative to the robot coordinate frame. This point is measured using the Microscribe so its position in robot coordinates is known and is called ^{R}p.

2. The end-effector is positioned such that this point can be measured by the vision system. This observed point will be referred to as ^{C}p, in the camera frame.

3. The end-effector is moved -100mm along the x-axis of the robot frame (pulling back

away from the point, ${}^{C}p$) and the point is measured by the vision system, ${}^{C}x$.

4. The change in coordinates, ${}^{C}x - {}^{C}p$, is divided by its magnitude to result in a unit vector, X, along the robot x-axis given in camera coordinates.

5. Steps 3 and 4 are repeated for the y- and z-axes to result in the corresponding unit vectors, Y and Z.

The transformation matrix, ${}^{R}T_{C}$, is constructed much the same as before. The unit vectors concatenate to form the 3×3 rotation matrix, ${}^{C}Rot_{R}$, as in Equation (3.22). The inverse of this, ${}^{R}Rot_{C}$, is used to align the coordinates of the original point, ${}^{C}p$, to the robot coordinate frame. The origin of the camera frame in robot coordinates is then the known position of the point, ${}^{R}p$, minus the vector from the camera frame origin to the observed point.

$$ {}^{R}C = {}^{R}p - {}^{R}Rot_{C} * {}^{C}p \tag{3.28} $$

Since the position of the toolplate centre is known when the observed point is measured in its original position, Equation (3.26) is used to find the transformation from the camera frame to the toolplate. This allows Equation (3.26) to be used when the robot is in any position, i.e. when ${}^{f}T_{C}$ is known, ${}^{R}T_{C}$ can be found for any ${}^{R}T_{f}$.

This is performed in software using another Mathscript node in Labview. This contains the transformations ${}^{R}T_{C}$ and ${}^{f}T_{C}$. The current position of the toolplate centre is passed in as is the matrix of points to be transformed. The node also has a Boolean input that is set by the user. This controls whether the static matrix ${}^{R}T_{C}$ is used or if it should be calculated as per Equation (3.26).

3.2.5 Vision System Integration

At this point the vision system was integrated with the hardware control system. Instead of measuring teat coordinates with the Microscribe, converting them into the robot co-

ordinate frame and inputting them manually into the top-level actuation control VI, the user would click on the endpoints of the teats and the robot would retrieve the cups and place them on the teats.

This was done by adding the top-level hardware control VI into the vision system block diagram. A cleaner top-level VI would have been obtained by placing the vision system and the actuation system in sequence in a new block diagram, however this was not possible. The image panes need to be on the top-level front panel but since they are inside a while loop, a sizeable portion of the vision system software must also be inside the loop and hence in the same VI. A sequence structure is created with the vision system software in the first frame. This means that these operations will be carried out before those in any other frame. The four teat coordinates are found in pixel coordinates and passed to the next pane. Here, triangulation is performed and the resulting coordinates transformed into the robot frame. Figure 3.16 shows the section of the block diagram where triangulation and transformation are performed and where the 3D coordinates of the teats in the robot coordinate frame are sent to the actuation system, the last block on the right. The vertical grey bars are the divisions between frames.

The result is a Labview VI that shows the user a video feed for each camera. It is not necessary to save images so there is only one image pane per camera. The user selects the endpoints of the teats in each feed using the mouse. The combined robot/end-effector actuation system then moves to the milking cup stand and retrieves the four cups simultaneously. These are brought into position beneath the teats and then applied. This was found to execute very reliably with no user interaction after the teats are selected onscreen. Although this did not provide data on the exact accuracy of the system it was seen as a major breakthrough, since the automated retrieval and application of the cups was a declared goal of the project.

For the testing phase, the control system was modified so that the cameras could be mounted on the end-effector and the teats measured at intervals as the end-effector approached them. To do this, the part of the block diagram concerned with acquiring

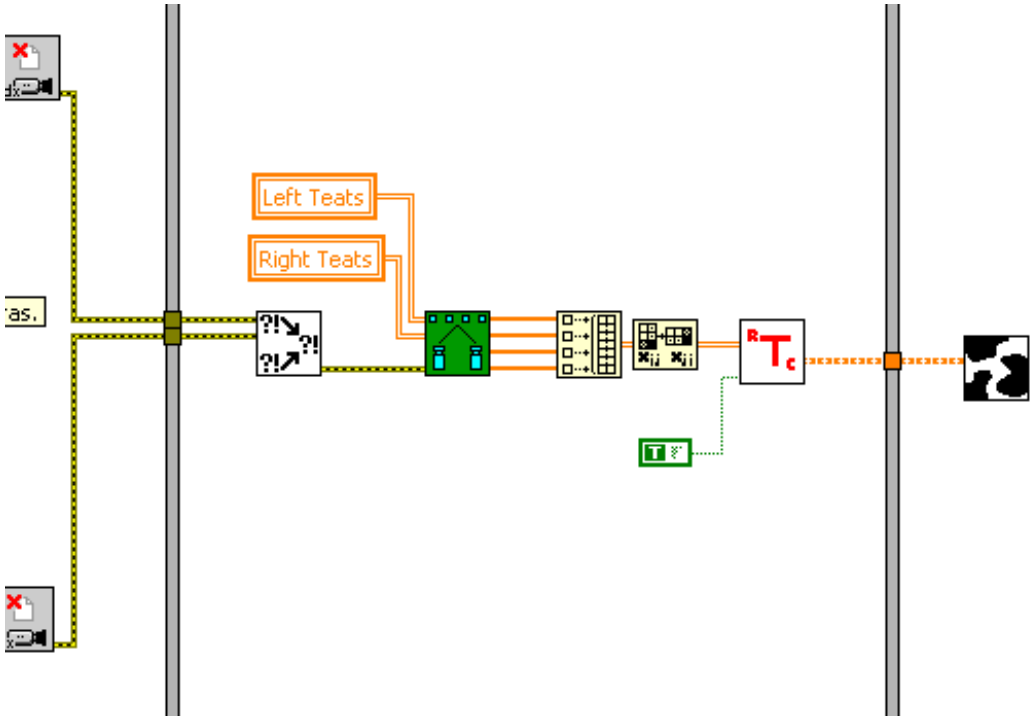


Figure 3.16: Triangulation and transformation of pixel coordinates.

the 3D coordinates of the teats in the robot frame was placed into a while loop. This while loop called the VI responsible for controlling the movement of the robot at each execution. This moved the robot towards the teat cluster by a certain pre-defined amount each time. Figure 3.17 shows how the robot position is incremented based on the loop iteration count. After executing 5 times the while loop terminates and the programme progresses. Five movements were chosen to give enough data points to evaluate the performance of the vision system. Higher resolution was not deemed necessary at this stage.

A section of the diagram is shown in Figure 3.18. This is basically the same section of the programme as that shown in Figure 3.16, but expanded to record certain values from the process to Excel sheets. This is also within the while loop so at every move, the data is recorded.

Another version of the software that was used in testing incorporated the Kalman filter (discussed in Section 5.5). This executed the filter in a Mathscript node. Because the Kalman filter recursively updates between iterations values from previous passes through

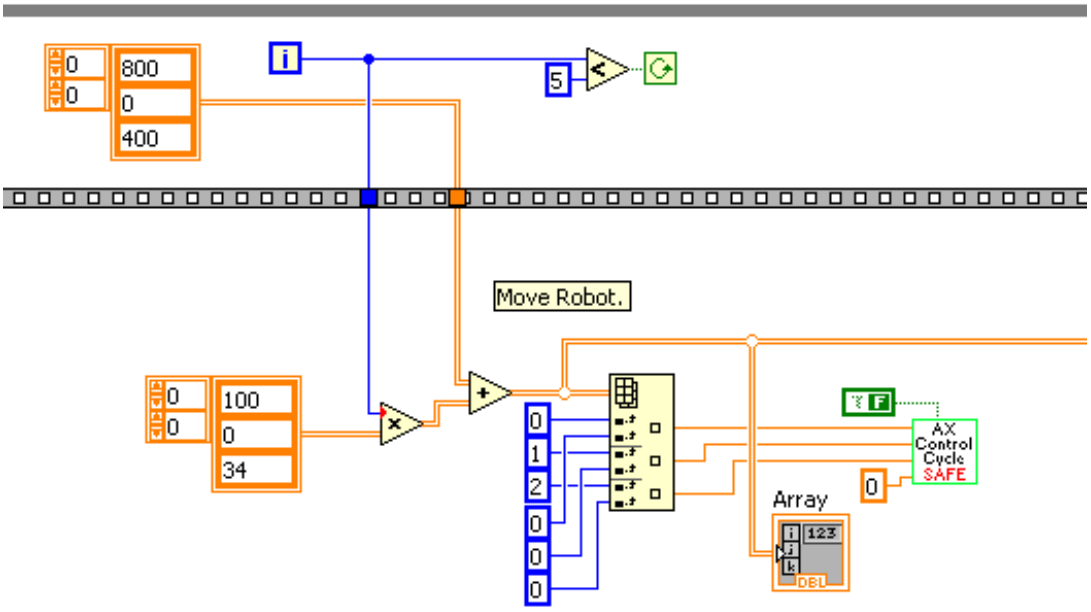


Figure 3.17: Incrementally moving the robot.

the loop were made available in the current step through the use of shift registers, which are optional terminals on the Labview while loop. This was inserted into the same sequence frame as the triangulation and the data was recorded a frame later. Again, the sequence was contained within the while loop. Different versions of the Kalman filter VI executed the loop for 6, 21 and 101 iterations. The final phase of testing, where the handling of misidentified teats was examined, used a similar programme to the Kalman filter setup in that the algorithms developed and simulated in Matlab were applied using a Mathscript node. This node was in the same place as in the Kalman filter programme but with minor changes to what data is passed to it.

These examples of interlinking Matlab scripts with Labview block diagrams highlight why these software environments were chosen. It provides a level of flexibility that combines the strengths of both environments. Matlab is designed for maths-intensive scripting that would result in complex block diagrams if it was attempted in Labview. Labview on the other hand can interface easily with hardware and allows for fast prototyping representing control flow in a very intuitive manner.

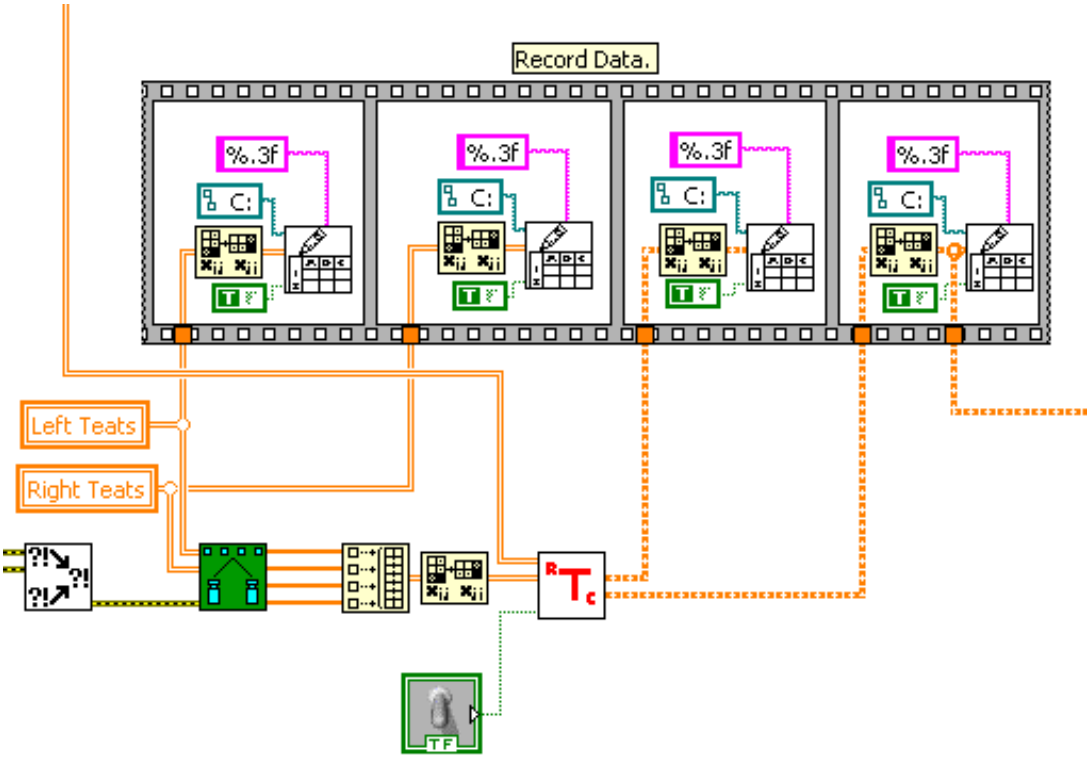


Figure 3.18: Triangulation and transformation of points as well as data storage.

Chapter 4

Data Estimation

4.1 Introduction

In such an unstructured environment as a dairy parlour, where the goal of the system is to physically interface with the teats of a cow, sensing becomes incredibly important in order to guarantee the safety of the cow and to successfully achieve the goal. The required level of accuracy can be achieved by the vision system alone, as proven in the lab by the successful application of milking cups to artificial teats. However when working with a live cow the system will be interacting with a dynamic environment and the teat positions may change at any moment. In such a situation, measurement noise will become more of a critical issue. While further research will result in the reduction of errors caused by inaccurate calibration and errors in transformation matrix estimation, the use of automated teat identification by the vision system may introduce a degree of error. It is thus prudent to investigate techniques that can use what is known about the system to make a more accurate estimation than is possible with the measurements alone.

4.2 Modelling

4.2.1 Theory

The additional information available comes in the form of a system model. This is a mathematical representation of the experimental setup. The experimental setup itself must be able to position the teats at predetermined locations in order to verify measurements and estimates. To do this, the plate to which the teats are mounted is attached to a screw-adjusted slider. This includes a scale that allows the slider to be adjusted to the nearest millimetre. This means that the teats can move in unison (individual movements would require a slider on each teat) in a straight line. In testing, the positions of the teats were shifted at by the same amount between each successive image giving a constant velocity. This means that in this case, the system model is not required to be a kinetic model. If acceleration were to be modelled, the result would be fewer data points. Also, since real data is not available, any acceleration value would be arbitrary and would bear no relation to the actual pattern of movement of the teats of a cow. In reality, the movement of the teats will depend on several factors e.g. the dimensions of the udder, the agitation of the cow, the pliability of the udder. While some of these factors cannot be modelled mathematically, some can. However there is very little measured data available regarding the motion of the udder and the factors that affect it. The acceleration then is chosen to be zero since a best case scenario is assumed in which the cow is not moving relative to the stall. In a working dairy parlour, the cow would have a velocity due to the motion of the stall on the carousel. The experimental setup in this work however, does not include the rotational motion and this will be added to the model at a later stage.

If the current position and velocity of an object is known, then it is possible to make an estimate of its position after a given time interval. The equations that describe the motion of a teat lead us to a system model. These equations of motion are presented in

Equations (4.1) - (4.2).

$$x_k = x_{k-1} + \dot{x}_{k-1}dt + \frac{1}{2}\ddot{x}_{x-1}dt^2 \tag{4.1}$$

$$\dot{x}_k = \dot{x}_{k-1} + \ddot{x}_{k-1}dt \tag{4.2}$$

In these equations, x represents a changing variable, in the case of this system, this is the position of a point along the axis of a coordinate frame. The subscript, k, indicates the time-step, with k being the current one. A dot above a variable indicates the first derivative, two dots indicates the second derivative. Equation (4.1) gives the position and Equation (4.2) gives the velocity. In the interests of reducing system complexity, and hence computational cost, the acceleration of the system is not modelled. The length of time between k and k-1 is given by dt. The above equations are converted into a discrete state-space equation as:

$$\begin{bmatrix} x_k \\ \dot{x}_k \end{bmatrix} = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_{k-1} \\ \dot{x}_{k-1} \end{bmatrix} + \begin{bmatrix} \frac{dt^2}{2} \\ dt \end{bmatrix} \ddot{x}_{k-1} \tag{4.3}$$

Since the second derivative dynamics of the system (the acceleration) are not being modelled, we can let the second term on the right hand side equal zero. However, if it was possible to control the movement of the system, it would be modelled as an acceleration similar to this term. Although in teat tracking there will be no control over the movement of the teat, it will provide a better understanding of the system if this term is left in, substituting the control input, u_{k-1}, for the acceleration of the system, $\ddot{x}_{k-1}$.

To introduce shorthand, the left hand side of Equation (4.3) will be represented by x_k. This is known as the state vector. The other substitutions are as follows:

$$A = \begin{bmatrix} 1 & dt \\ 0 & 1 \end{bmatrix} \tag{4.4}$$

$$B = \begin{bmatrix} \frac{dt^2}{2} \\ dt \end{bmatrix} \tag{4.5}$$

This results in Equations (4.3) being represented as:

$$x_k = Ax_{k-1} + Bu + w_{k-1} \tag{4.6}$$

This is known as the state transition equation. A third term has been added in to account for deviations from the system model between states. This is called process noise, w_k, and is assumed to be Gaussian with zero mean and covariance, R_k.

Since the state cannot be directly observed, the measurement is given as a function of the current state with added noise, v_k. This measurement noise is again Gaussian with zero mean and covariance, Q_k.

$$y_k = Cx_k + v_k \tag{4.7}$$

where $C = \begin{bmatrix} 1 & 0 \end{bmatrix}$, indicating that the sensor only measures position. Equations (4.6) and (4.7) describe a system that is linear and Gaussian.

4.2.2 Simulation

Since this model will be incorporated into data estimation techniques it must be evaluated itself before attempting to evaluate and compare the techniques used. The system model must match as closely as possible the experimental setup in order to get a realistic idea of the performance of the data estimation techniques. To evaluate the system model expressed in Equations (4.6) and (4.7), the measurements produced by it were compared to those taken from the experimental system in the lab. The teats were moved using the slider in increments of 1mm. By moving to the next millimetre mark and not 1mm from the current position, the error is not allowed accumulate. This is expressed in the system model by setting the second term in the B matrix to zero. This tells the system that the

process noise affects the position state but not the velocity state.

The process noise acts on the system as an acceleration. It does this by adding a random value, with a certain standard deviation, σ, to each state through the B matrix. Thus, the noise term in Equation (4.6) is implemented as:

$$w_k = B * \sigma * rand \tag{4.8}$$

where $rand$ is a normally distributed random vector and σ is the standard deviation. Since the covariance, R_k, is given as:

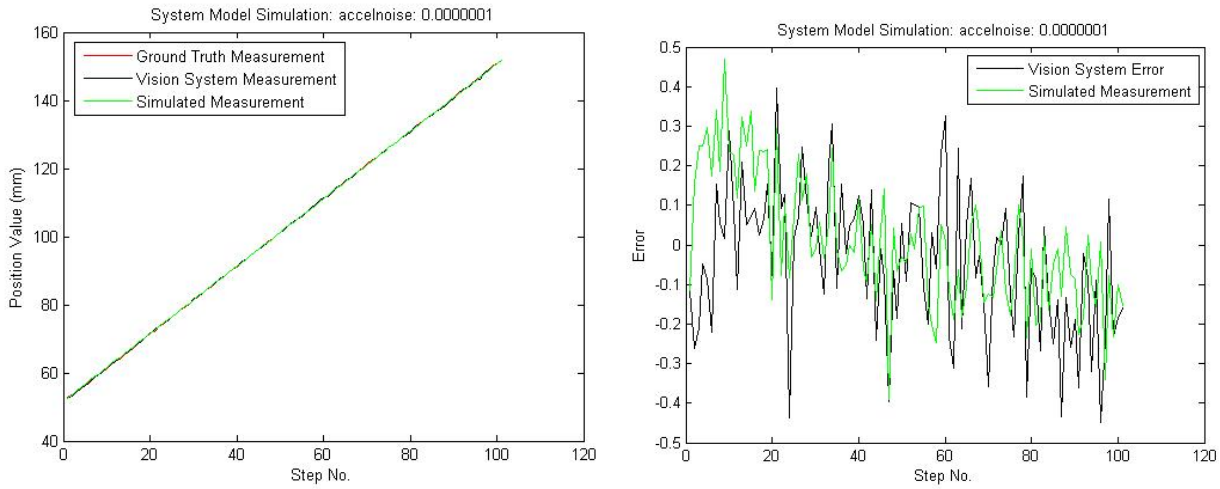
$$R_k = E[w_k {w_k}^T] \tag{4.9}$$

where $E[x]$ is the expected value of x, R_k is then given as:

$$R_k = (B * \sigma)(B * \sigma)^T \tag{4.10}$$

This is the formulation of the process covariance matrix used in [54] and is one of several tested in Section 5.5.

Figure 4.1a shows the results of the simulation compared to the positions measured by the Microscribe and by the vision system. The simulation is initialised with a position equal to that of the first vision system measurement. The velocity is set to give as close an approximation as possible to the ground truth measurements (0.99mm per iteration). For this simulation the measurement noise standard deviation is given as 0.2mm (from experimental data) and the process noise standard deviation is set to 0.0000001mm/s^2, i.e. near zero. Figure 4.1b shows the error values of the vision system and the simulation compared to the Microscribe. This shows the velocity is accurate, with a lower error variance. This is as expected since the process noise was set so low.



(a) Position

(b) Position Error

Figure 4.1: Simulated measurements using modified process noise standard deviation.

4.3 Bayes Filter

There are now two sources of information available about the system: the sensor measurements, and the system model, which can also provide expected measurements. Neither of these sources is completely accurate and has an associated error probability density function. A technique is required then that will combine these sources with some improved degree of accuracy.

Such a technique is found in the recursive Bayesian estimator. This estimates the states of a system using measurement data as well as data from the model of the system. The Bayes filter, as the recursive Bayesian estimator is also known, makes use of the Markov assumption that the future and past states are independent if the current state is known [21]. (A series of such states is called a Markov chain). This means that the probability of the current state depends only on the last state and not on any states prior to that. This also introduces a limitation in that the assumption can be violated. Unmodelled system dynamics and errors in process and measurement noise parameters can cause violations of the Markov assumption. The Bayes filter takes the probability of the state having a certain value at the last time-step, and predicts the current value, x_k, of the state given the control input, u_k. This is shown in Equation (4.11). The

result is known as the a priori estimate. The second step (Equation (4.12)) then uses the current measured value of the state and combines it with the a priori estimate. The result is called the a posteriori estimate and represents the probability distribution of the estimated value of the state, integrating the system model estimate and the measurement. In these discussions, the subscript k refers to the current time-step and k-1 refers to the last time-step.

Prediction step:

$$p(x_k|Y_{k-1}) = \int p(x_k|u_k, x_{k-1})p(x_{k-1}|Y_{k-1})dx_{k-1} \tag{4.11}$$

Update step:

$$p(x_k|Y_k) = \eta p(y_k|x_k)p(x_k|Y_{k-1}) \tag{4.12}$$

In these equations $p(x_k|Y_k)$ represents the probability distribution of the estimate, given all measurements up to the current time and is often referred to as the posterior. η is the normalization function. Upper case Y_k indicates all measurements up to time k. The Bayes filter is discussed further in [21], [55] and [30].

4.4 Kalman Filter

4.4.1 Theory

The Kalman filter is a Bayes filter that deals with linear Gaussian systems. This means that the state transition equation and the measurement equation are both linear with added Gaussian noise and the a posteriori estimates are represented as a Gaussian distribution. The algorithm is presented in Equations (4.13) to (4.17) below.

$$\hat{x}_k = A_k x_{k-1} + B_k u_k \tag{4.13}$$

$$\hat{P}_k = A_k P_{k-1} A_k^T + R_k \tag{4.14}$$

$$K_k = \hat{P}_k C_k^T (C_k \hat{P}_k C_k^T + Q_k)^{-1} \tag{4.15}$$

$$x_k = \hat{x}_k + K_k(y_k - C_k \hat{x}_k) \tag{4.16}$$

$$P_k = (I - K_k C_k)\hat{P}_k \tag{4.17}$$

In these equations, A_k, B_k and C_k are the current values of the A, B and C matrices in Equations (4.6) and (4.7). Using the k subscript allows the system equations to change but this will not be modelled in this application and it is included only for completeness. Equation (4.13) gives the a priori estimate. This is the estimate based only on the system model's state transition equation (Equation (4.6)) and serves as the prediction of the current state. The next line gives the covariance of this prediction and integrates the covariance of the process noise. These two equations represent the prediction step of the Bayes filter. In Equation (4.15) the Kalman gain is calculated. The Kalman gain is used (in Equations (4.16) and (4.17)) to augment the a priori state estimate and covariance with an amount proportional to the actual measurement, y_k, and the measurement calculated by the system model. The Kalman gain is formed such that it is inversely proportional to the measurement noise covariance, Q_k, and proportional to the process noise covariance, R_k, through the a priori estimate covariance. This means that if the measurement noise is large K_k will be small and the measurement, y_k, will contribute less to the a posteriori estimate. If the process noise is larger, K_k will be larger and the measurement will have a greater influence. Equations (4.16) - (4.17) represent the update step of the Bayes filter. The Kalman filter minimizes the mean squared error of the estimate [56] and provides the optimal solution to the estimation problem [57].

The Kalman filter is derived in full in [58].

4.4.2 Simulation

The Kalman filter was simulated in a Matlab script. The Kalman filter algorithm, presented in Equations (4.13) to (4.17), is easily expressed in Matlab. At this stage the purpose was to study its operation. Accurately modelling the real system would come later but for now the simulation was made as simple as possible. This meant tracking a single variable changing according to Equation (4.6) with u = 0, that is, no control input. The variable was initially set to have position and velocity values of zero. The filter was given these as its initial state estimates, i.e. x_0. It was also given an identity matrix for the initial estimate covariance, P_0. This tells the filter how much the initial state estimate can be trusted. In the application of this filter (see Section 5.5) the movement was known to be predominantly in a direction known to have a measurement error standard deviation of approximately 0.2mm. This was also input to the simulation. The initial values of the states, the state estimates, the state estimate covariance and the measurement noise covariance were thus prescribed for the simulation. The only remaining parameter was the process noise covariance. According to [56], the process noise covariance can be tuned to give better performance. In this simulation we are specifying the process noise in the system. We are also using it to calculate the Kalman gains. In a real-world application, the system will have an inherent, unknown process noise and the parameter used in the filter will be an estimate. In the simulation however, since the filter uses the exact process noise of the system, the best performance results when the process noise is at its smallest value, i.e. when the system exhibits least unpredictability. Figure 4.2 shows the results of the simulation. The simulation is run for 100 iterations and these correspond to the horizontal axis of the plots. The vertical axis gives the errors between the measurement values and the actual states (black line), and between the Kalman filter state estimates and the actual states (green line). Figure 4.2a uses a value of 0.1 for the process noise standard deviation and Figure 4.2b uses a value of 0.0000001. The filter obviously per-

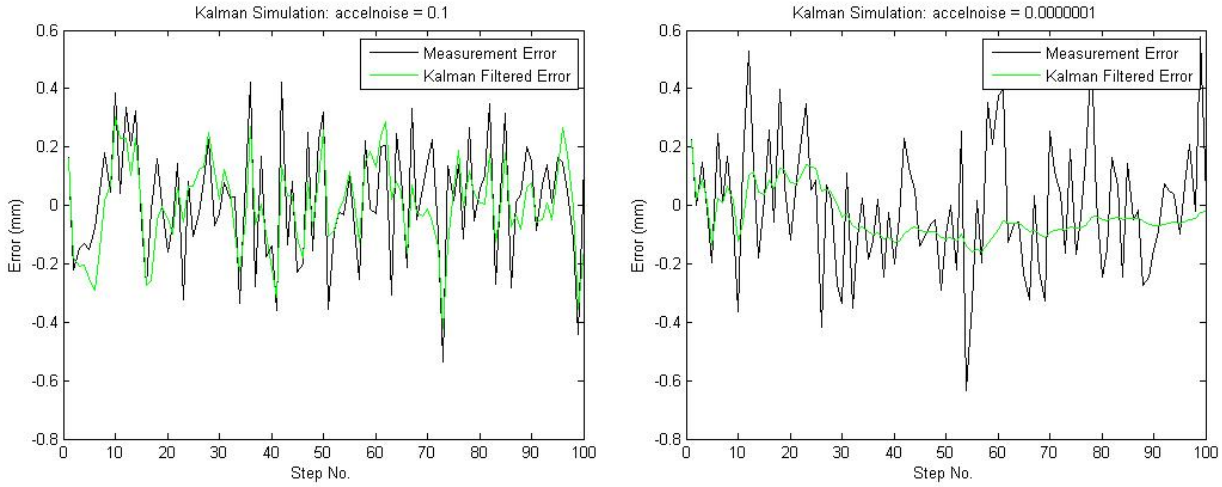
forms far better in the second case. What it is essentially being shown is that the system model has very low noise and is more accurate as a result. Less importance is then placed on the measurement values. This is reflected in the Kalman gains from each run of the simulation. With process noise standard deviation at 0.1, the Kalman gains are:

$$K_k = \begin{bmatrix} 0.6284 \\ 0.3048 \end{bmatrix} \tag{4.18}$$

When this value is set equal to 0.0000001:

$$K_k = \begin{bmatrix} 0.0394 \\ 0.0006 \end{bmatrix} \tag{4.19}$$

Referring back to Equation (4.16), when the Kalman gain is large, the measurement contributes more to the update.

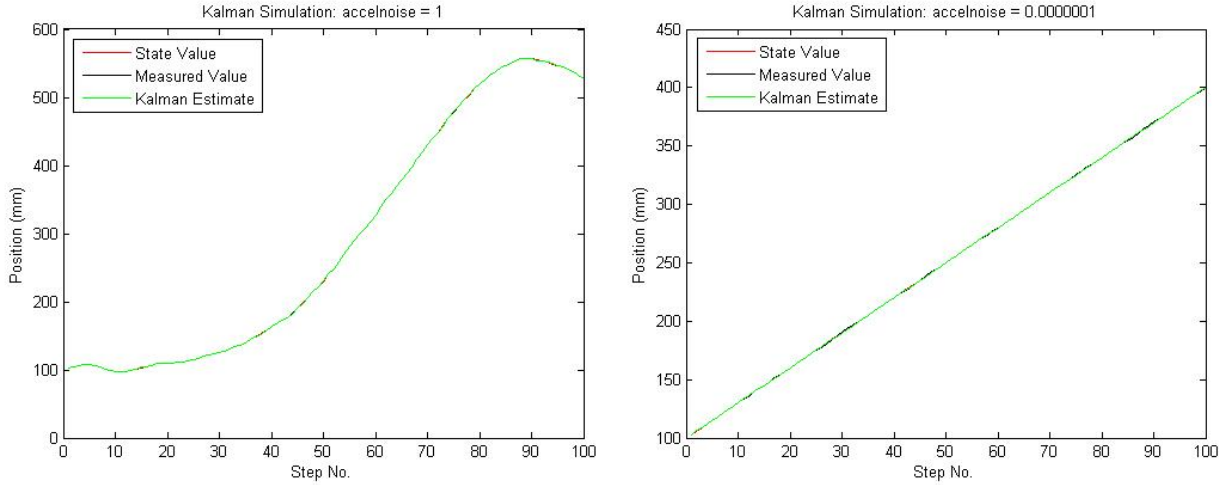


(a) Process Noise, $\sigma = 0.1$ (b) Process Noise, $\sigma = 0.0000001$

Figure 4.2: Simulation error results.

Varying the process noise standard deviation does not only change the Kalman filter state estimates. It changes the actual state transitions so while reducing this value to near zero may improve things drastically, it is actually because there is less variance in the system. Figure 4.3a shows an extreme case of this with this value set to 1. Figure

4.3b shows the previous value of 0.0000001. The larger process noise causes rapid changes in position that are not seen with the smaller value. This illustrates a major limitation of the simulation. On one hand the process noise is used to define the system, on the other, it is used to estimate it.

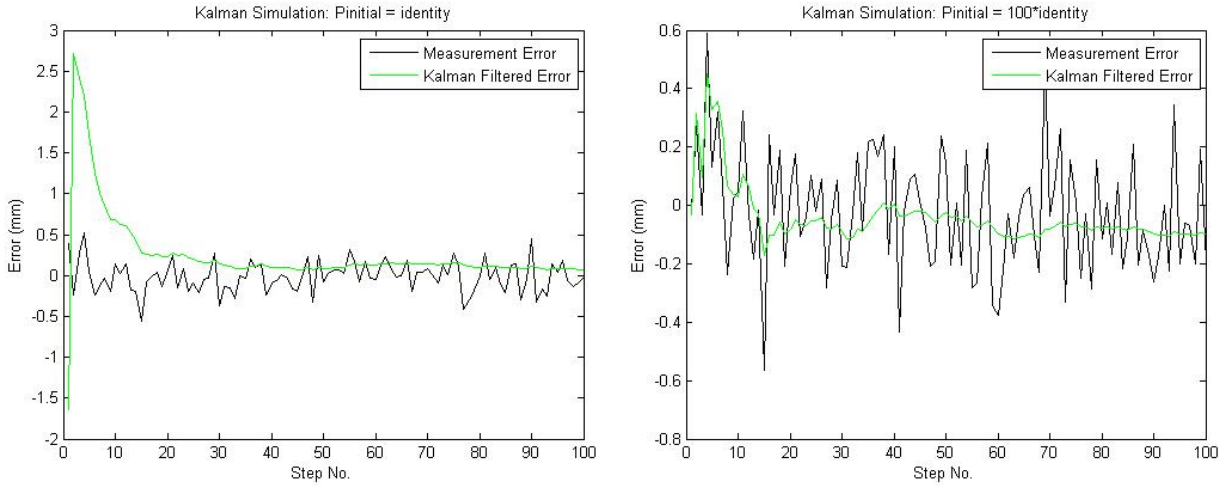


(a) Process Noise, $\sigma = 1$ (b) Process Noise, $\sigma = 0.0000001$

Figure 4.3: Simulation position results.

The Kalman filter estimates can be perturbed by giving an initial state that differs from the initial state estimate. Figure 4.4 show the results of the simulation with the process noise covariance the same as in Figure 4.3b and with the initial states set to a position of 100mm and a velocity of 3mm/iteration. (The time interval value, dt = 1, is used for all of these simulations). In Figure 4.4a the initial value for the estimate covariance matrix, P_0, is left as the identity matrix. This tells the filter that the initial state estimate is trustworthy. In this run of the simulation the initial state estimate is still set to zeros. It thus takes several iterations for the filter to realize that the state estimates are not trustworthy. In Figure 4.4b the Kalman filter estimate error follows that of the measurements closely for the first few iterations. This is because it has been told not to trust the initial estimate and so it follows the measurements more closely until it finds the correct value.

The next modification to the simulation is to use realistic initial states and to set

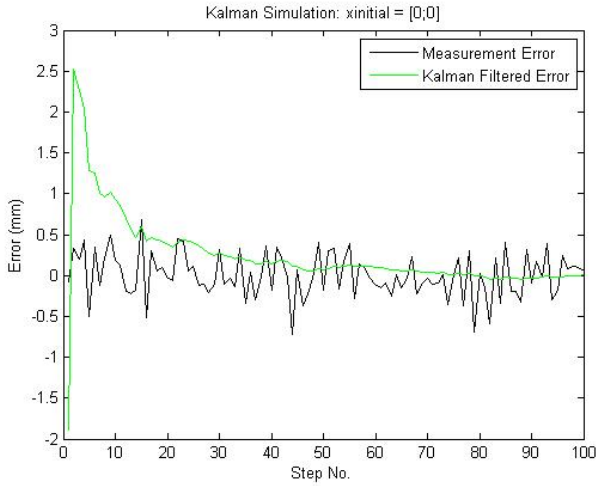


(a) Initial Estimate Covariance, $P_0 = \mathrm{I}$ (b) Initial Estimate Covariance, $P_0 = 100 * \mathrm{I}$

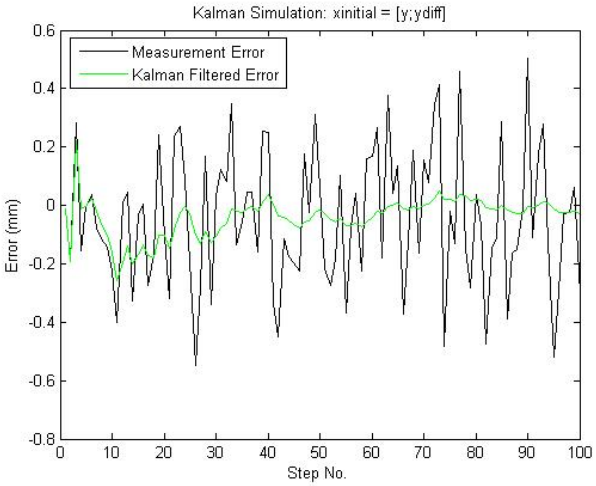
Figure 4.4: Simulation error results.

the initial state estimates to more intuitive values. The results to these simulations are shown in Figure 4.5. Plot (a) shows initial state estimates set to zeros, the estimate covariance set to the identity matrix and the process noise standard deviation set to 0.0000001 as before. The initial position value was set to 52.65mm and the velocity to 1mm per iteration. This data is arbitrary and based on the idea that the teats will be tracked in a coordinate frame with the origin at the centre of the teat cluster. In Figure 4.5b, the Kalman filter doesn't begin estimating until the second iteration. On the first iteration the estimate is given as being equal to the first measurement. On the second iteration when the filter makes its first estimate, the initial state estimate is taken as being equal to the current measurement for the position and the difference between the first two measurements, ydiff, for the velocity. Although not very accurate, this is visibly an improvement over the performance shown in Figure 4.5a. The initial estimate covariance is set to the identity matrix in both simulations.

To quantify the improvement, the standard deviation over the last 50 iterations (to exclude any settling time) is reduced from around 0.2mm in the raw measurement values to around 0.02mm, or sometimes even less, for the Kalman estimates. An issue arises when the simulation is run for far longer. At 1000 iterations, there appears to be a far



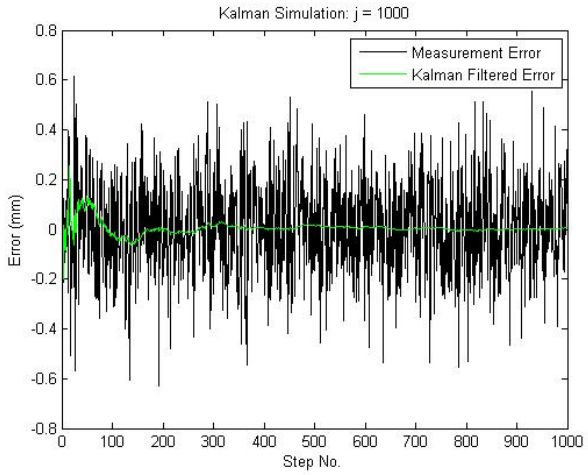
(a) Initial State Estimate, $x_0 = (0; 0)$



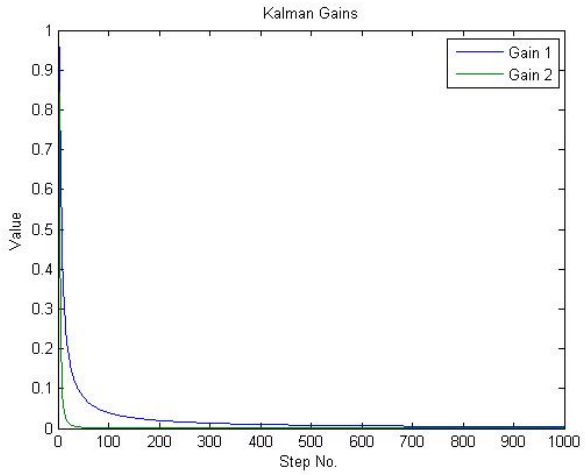
(b) Initial State Estimate, $x_0 = (\ y_2; y_2 - y_1\ )$

Figure 4.5: Simulation error results.

longer settling time. This is shown in Figure 4.6a. This shows that while the performance does improve in the first few iterations, as highlighted by the standard deviations, if left to run, it will improve even more.



(a) Error Results



(b) Kalman Gains

Figure 4.6: Simulation results.

Figure 4.6b shows the convergence of the Kalman gains over the course of the simulation. From Equation (4.16), the first element in the Kalman gain matrix adjusts the estimate of the first state and the second gain element adjusts the second state estimate. So looking at Figure 4.6b we see that when the gain associated with the velocity reaches

steady state, there is a large improvement in the performance. The gain associated with the position however takes a lot longer to reach steady state and it is not until it does, that the errors on the Kalman estimate errors reach near zero levels (standard deviation of 0.0018mm with mean 0.0021mm over the last 100 iterations). The reason the errors become so small is because since the gains go to near zero, the Kalman filter is relying nearly completely on the system (a priori) estimates. The velocity is hardly changing and the system dynamics become very predictable. Once the first gain reaches steady state, the filter parameters don't need to change any more.

It was decided to simulate a steady-state Kalman filter. The idea of this is to use the steady-state Kalman gains and state estimate covariance from the start of the run. The advantage of this, is that the execution time of the filter is reduced. This is discussed [59] where the steady-state gains are applied to a disk drive. This is expected to provide results similar to what is seen in the final few hundred iterations in Figure 4.6, i.e. errors with very low mean and standard deviation. There are two ways to obtain the steady-state gains. The most obvious is to use those resulting from a run of the time-variant filter. This is described in [59], where the suggestion is to run an initialisation phase at start-up. This way, the Kalman gains and estimate covariance can be found and settling time of the filter can be avoided. Tzafestas discusses this in reference to a hard disk drive head positioning actuator. This is a fairly structured environment. Whether the technique could be applied to a more chaotic environment like a dairy parlour remains to be seen. Another method of finding the steady-state Kalman gains and estimate covariance is discussed in [60]. Here, the state estimate covariance is found by solving the algebraic Riccati equation:

$$P = APA^T - APC^T(CPC^T + Q)^{-1}CPA^T + R \tag{4.20}$$

This is easily done in Matlab using the dare() function which accepts the A, B, Q and R matrices as arguments and returns the steady state estimate covariance matrix.

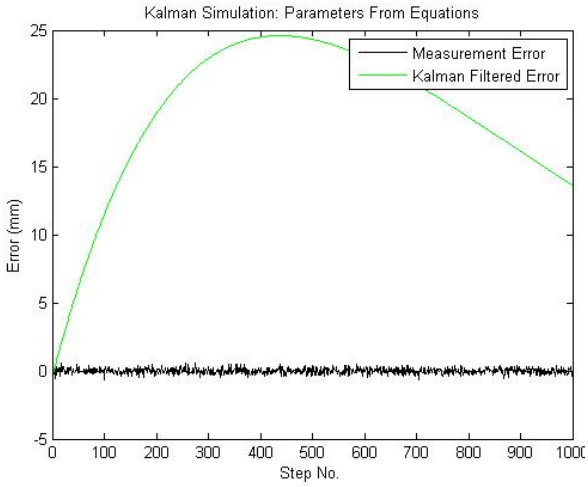
Equation (4.21) is used to find the steady-state Kalman gains.

$$K = PC^T(CPC^T + Q)^{-1} \tag{4.21}$$

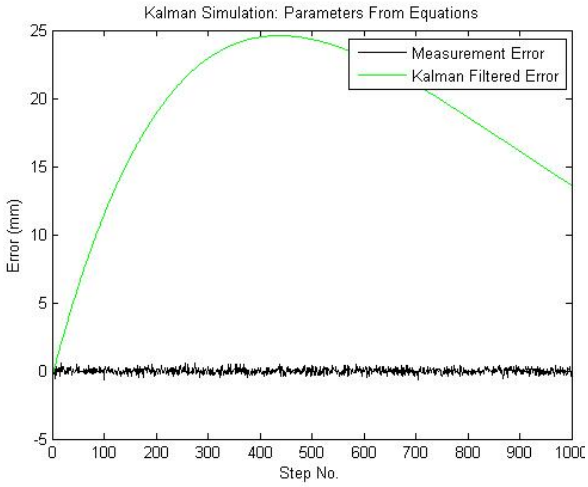
Equations (4.20) and (4.21) are also given in [61]. The results of simulations using these techniques are shown in Figure 4.7. Both show a similar pattern. Repeated runs of the simulations both showed overshoots of varying magnitude and sign with little or no difference in behaviour between the two techniques. The Kalman gains and estimate covariances for each are shown in Table 4.1. The behaviour of the Kalman filter is typical of values of the estimate covariance that are too low, as was illustrated in Figure 4.4.

Table 4.1: Steady state parameters calculated using the two methods of finding the steady-state values.

	Kalman Gains	Estimate Covariances
Tzafestas	$\begin{bmatrix} 0.004 \\ 6.02 \times 10^{-6} \end{bmatrix}$	$\begin{bmatrix} 1.6 \times 10^{-3} & 2.41 \times 10^{-7} \\ 2.41 \times 10^{-7} & 4.85 \times 10^{-10} \end{bmatrix}$
Newland/Gray	$\begin{bmatrix} 0.004 \\ 5.99 \times 10^{-6} \end{bmatrix}$	$\begin{bmatrix} 7.07 \times 10^{-12} & 1 \times 10^{-8} \\ 1 \times 10^{-8} & 2.83 \times 10^{-5} \end{bmatrix}$



(a) Tzafestas method



(b) Newland/Gray method

Figure 4.7: Simulation error results.

The Tzafestas method was tried during testing on the experimental system. The limitations of the hardware system meant that it was difficult to collect many data points.

One run was completed using a time-variant Kalman filter and the Kalman gains recorded. These gains were then used in a steady-state Kalman filter in a second run. This test is discussed further in Section 6.5.2. The final Kalman gains found were then put back into the above simulation and the Kalman filter performance improved. In this simulation, the state estimate covariance was allowed to update. The result of this simulation is shown in Figure 4.8. It resembles the results seen later in testing where the Kalman filter improves performance to an extent as the test goes on but in the first few iterations the difference is ambiguous.

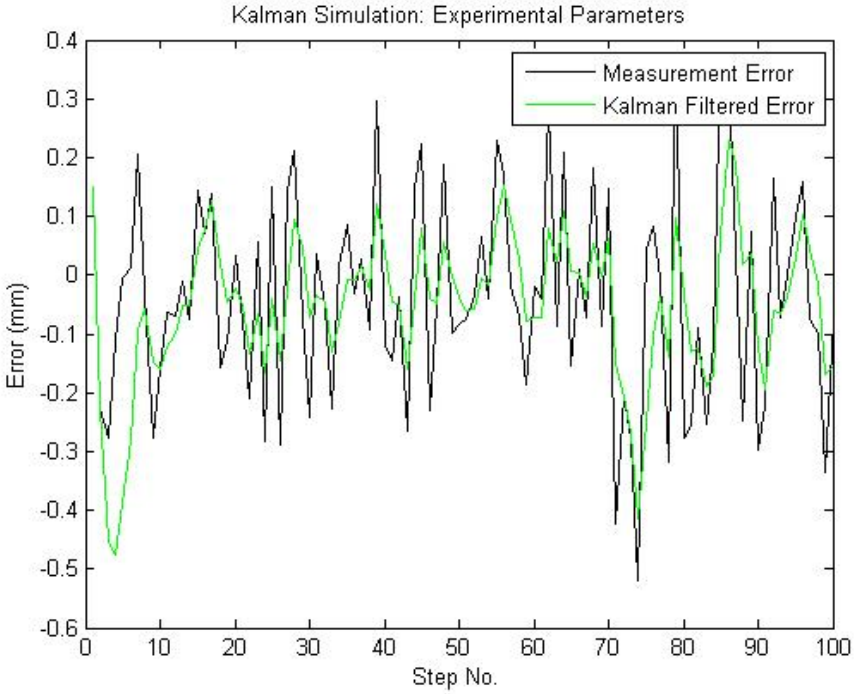


Figure 4.8: Simulated error results using experimentally found Kalman gains.

The Kalman filter shows huge potential for improving the measurements in this system. Indeed, even just looking at its popularity and range of applications shows that it should be well suited for use in the vision system of the milking cup application system. While the steady-state filter should give better results, it does not appear to do so using either of the theoretical methods of finding the steady-state parameters. The experimental method works better but does still not match the performance of the time-variant Kalman filter shown in Figure 4.5b. It must be noted that the performance of the filter

in simulation is expected to be different to what will be seen in laboratory tests. This is due to the noise covariances being unknown, unlike in these simulations where they are specified.

4.5 Particle Filter

4.5.1 Theory

Another type of Bayes filter that was investigated was the particle filter. This is a nonparametric filter meaning that the posterior is not modelled as a set of parameters (such as a Gaussian distribution) but as a set of values. These values can represent regions in state-space (as in the histogram filter) or points in state-space. The particle filter does the latter. As a Bayes filter, the particle filter includes an estimation step and an update step. A set of state hypotheses (particles) is drawn from the posterior. The distribution of these particles will approximate the posterior and will get closer as the number of particles approaches infinity. Each particle is propagated through the system equations, including adding process noise, to give a set of a priori estimates. An estimated measurement is generated for each particle. The probability of an estimated measurement being the actual measurement is calculated by incorporating the actual measurement and the measurement probability density function. This provides weights for each particle in the set. These weights are then used to define a distribution from which a new set of particles is drawn. This updated set of particles will be a set of samples distributed according to the new posterior. This idea of using a set of random samples to approximate a value over and over is known as a sequential Monte Carlo method.

Key to understanding the operation of the particle filter is the idea of Monte Carlo integration. This is used when we want to integrate a function $g(x)$, the target distribution, but we only have access to a distribution $\pi(x)$, the proposal distribution. What Monte Carlo integration does is to select a set of random points drawn from $\pi(x)$ and multiply it by a weighting function, $f(x)$, which represents the probability that the selected points

fall within $g(x)$.

$$I = \int g(x)dx \tag{4.22}$$

So, instead of sampling from it directly, we take samples from another function that we can sample from, $\pi(x)$. We are essentially factorising $g(x)$ such that:

$$g(x) = f(x).\pi(x) \tag{4.23}$$

The integration becomes:

$$I = \int f(x).\pi(x)dx \tag{4.24}$$

The Monte Carlo estimate of the integral is then the mean of the samples:

$$I_n = \frac{1}{N}\sum_{i=1}^{N} f(x^i) \tag{4.25}$$

where N is the number of particles and i is the index of the particles. Monte Carlo integration is discussed further in [62].

The principle of Monte Carlo integration is applied in the particle filter in the form of importance sampling. The target distribution, $g(x)$, is the posterior of the current iteration. The proposal distribution, $\pi(x)$, is the posterior of the previous iteration. The set of particles are drawn from this. The function $f(x)$ is a set of weights that relates the particle estimates (distributed according to $\pi(x)$) to the observations of the state, i.e. the measurements.

A more detailed pseudo code algorithm for the particle filter is presented below. This algorithm is taken from [32].

1. $particlefilter(X_{k-1}, u_k, z_k)$
2. $\hat{X}_k = X_k = 0$
3. $for\ m = 1\ to\ M$

4. *sample* $x_k^{[m]} \sim p(x_k|u_k, x_{k-1}^{[m]})$

5. $w_k^{[m]} = p(z_k|x_k^{[m]})$

6. $\hat{X}_k = \hat{X}_k + \langle x_k^{[m]}, w_k^{[m]} \rangle$

7. *end*

8. *for* $m = 1$ *to* M

9. *draw i with probability* $\propto w_k^{[i]}$

10. *add* $x_k^{[i]}$ X_k

11. *end*

12. *return* X_k

The filter creates a finite set of M particles initialised to a range of values spread over state-space. It then takes each particle and creates an estimate by propagating it through the system equations adding a process noise value. This is represented in line 4. The estimated measurement for each particle is also found, again adding a random value with variance equal to that of the measurement noise. In line 5, each particle is given a weight proportional to the probability of that particle being the correct estimate, that is, according to the probability distribution of the measurement error. In this way, the actual measurement value is incorporated, thus corresponding to the Bayes filter update step. In the set of particles, the estimates that are closer to the true value of the state end up with higher weights. These weights are normalised so that they sum to 1. This is so that they more accurately represent a Gaussian distribution, in that the integral of the distribution is 1. Lines 9 and 10 in the algorithm show the resampling step. This is where a new set of particles is selected. The new set has the same number of particles as before and these are taken from the old set. Particles with higher weights are more likely to be selected and may be selected more than once. The new set of particles represents

the posterior. Several different resampling algorithms are discussed and compared in [36]. Two algorithms are found to be favourable, stratified and systematic resampling. Stratified resampling is used here.

4.5.2 Simulation

The particle filter was again implemented in Matlab. Initially, the filter was performed on simulated data, as with the Kalman filter. A two element state vector was used corresponding to position and velocity. Arbitrary parameters were selected to be similar to those seen in the Kalman filter simulations, such as process and measurement noise covariances as well as initial states. The same system model was used as in the Kalman filter simulations with $u = 0$, that is, no control input.

The main difference in parameters that can affect the particle filter that are not present in the Kalman filter are the number of particles used and the initial distribution of the particles in state-space. As mentioned above when a set of particles is drawn from a distribution, the set of particles will have a distribution close to the original, but not exactly the same. As the number of particles goes to infinity, the difference between the two goes to zero as the distribution of the particles more fully describes the original distribution. Therefore, the more particles that are used in the filter, the more accurate it will become. This is not really practical however since the amount of calculations increases with the number of particles, i.e. a filter with 1,000 particles will take about 10 times longer to run than a filter with 100 particles. Thrun [32] states that the number of particles used is normally about 1,000. It is also stated that the difference between the original distribution and that represented by the set of particles is not an issue unless for very small numbers of particles, e.g. <100. In this simulation 1,000 particles were found to work well with performance deteriorating as the number is reduced.

The initial distribution of the particles in state-space did not appear to have much significance for the values used. The system was given initial position and velocity of -30mm and 1 mm/s respectively. The initial set of particles was distributed around

0mm and 0mm/s with a standard deviation of 10mm for the position and 5mm/s for the velocity. These values worked adequately and performance was not greatly affected until these numbers were brought down to below about 1mm and 1mm/s each. Figure 4.9 shows a typical set of results from the particle filter. In this, the standard deviation of the resulting measurement error is 1.0184mm. This is reduced to 0.6049mm by the particle filter. This is clearly a significant improvement.

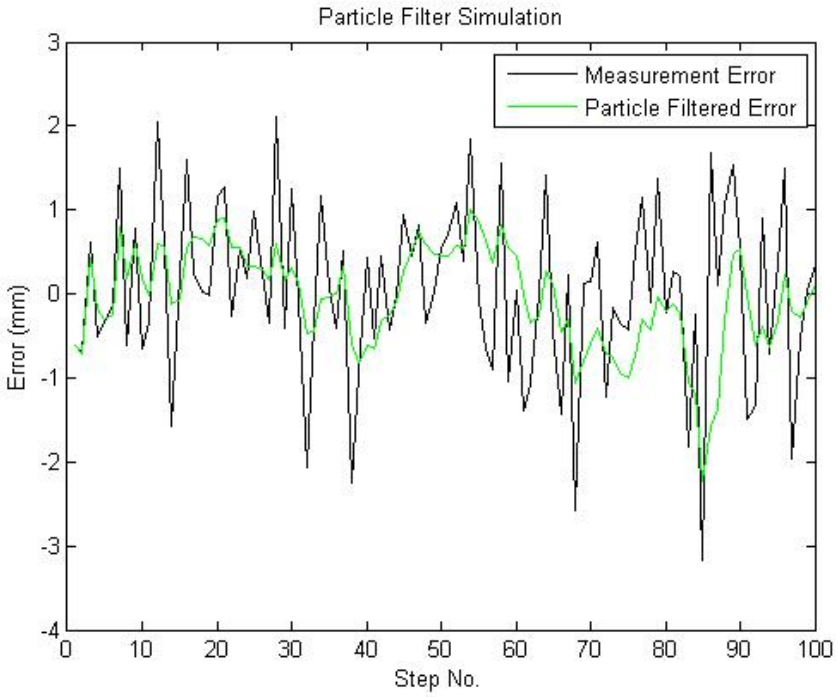
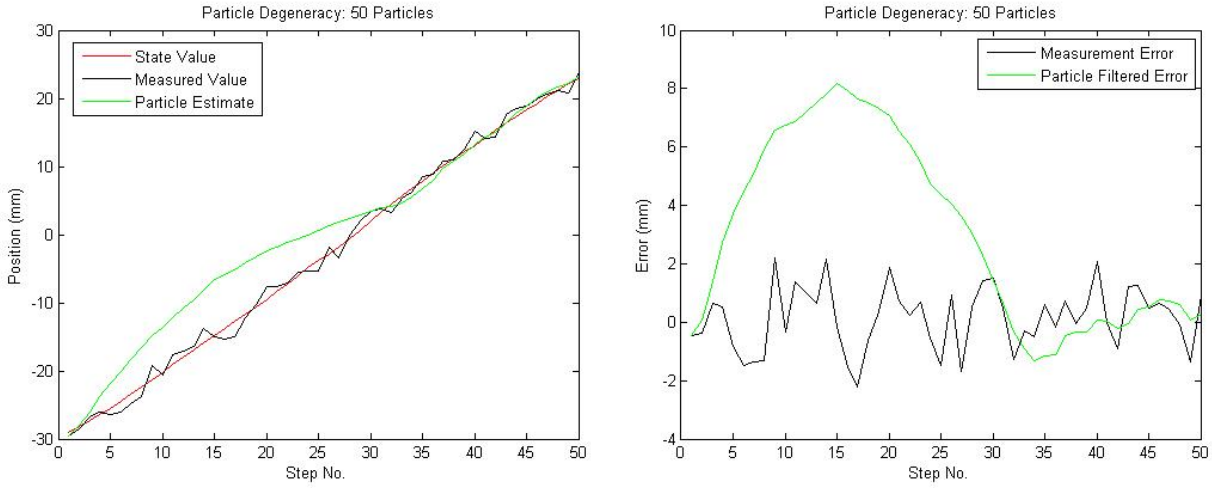


Figure 4.9: Simulation error results - particle filter.

For this simulation, the measurement noise standard deviation of the system was set to 1mm. The process noise was set to 0.1. Unlike the Kalman filter, the particle filter performs better when the process noise is not set too low. The reason for this lies in line 4 of the algorithm. When the set of particles is propagated through the system in the prediction step of the filter, process noise is added. Since this set of particles resulted from the previous iterations resampling stage many particles will be duplicates. This additive process noise serves to create a set of hypotheses about where the states are now. They need a certain amount of diversity or else the filter can become prone to the particle deprivation problem [63], also known as sample impoverishment [33] [64] or

degeneracy [34][35]. This is where the particles are not located particularly close to the true state and their weights reduce to near zero.

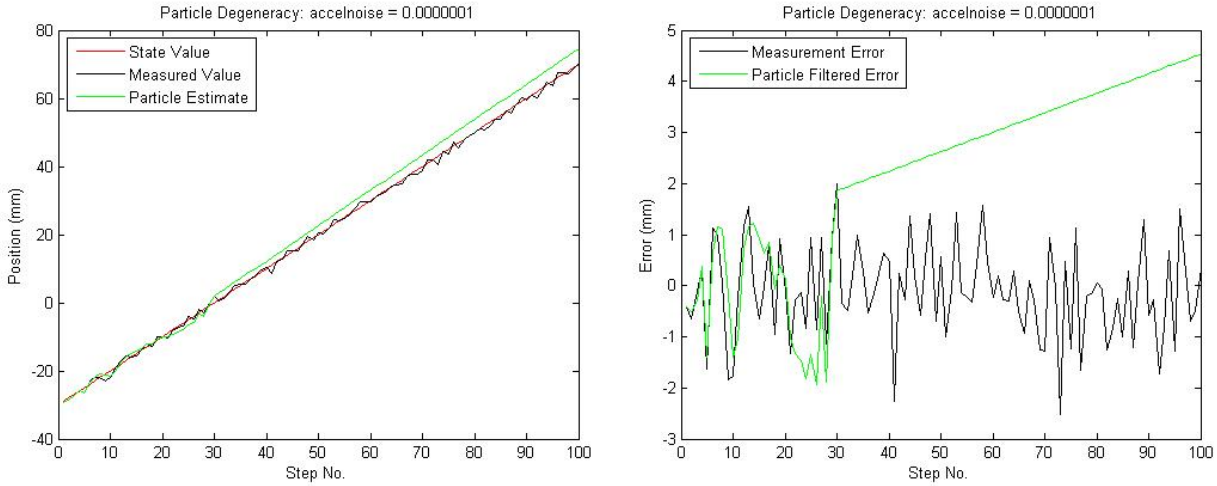
Degeneracy can occur for several reasons. If there are too few particles in a set, the likely state-space will not be sufficiently covered and there will be no hypothesis near the true state. This is illustrated in Figure 4.10 where the number of particles was set to 50. Another possible cause of degeneracy is the one that led us to this discussion: that of not using a large enough process noise variance. When this happens, state estimates represented by each particle will be very close to the resampled particles resulting from the previous iteration. This results in fewer hypotheses of the state, effectively reducing the number of particles. This is shown in Figure 4.11 where the process noise has been set to a very low value, 0.0000001. Finally, it is possible for degeneracy to occur as a result of bad luck. Since random numbers are used in the resampling process, it is possible that a series of them in the wrong place can cause the weights of a lot of the particles to go to near zero and again there will be fewer effective particles.



(a) Position (b) Error

Figure 4.10: Simulated degeneracy - 50 particles

Another way to disturb the particle filter is to set the measurement noise variance too low. This results in very low weights being chosen for most of the particles and these quickly go to zero. In the first two examples of particle deprivation, there have still been a



(a) Position

(b) Error

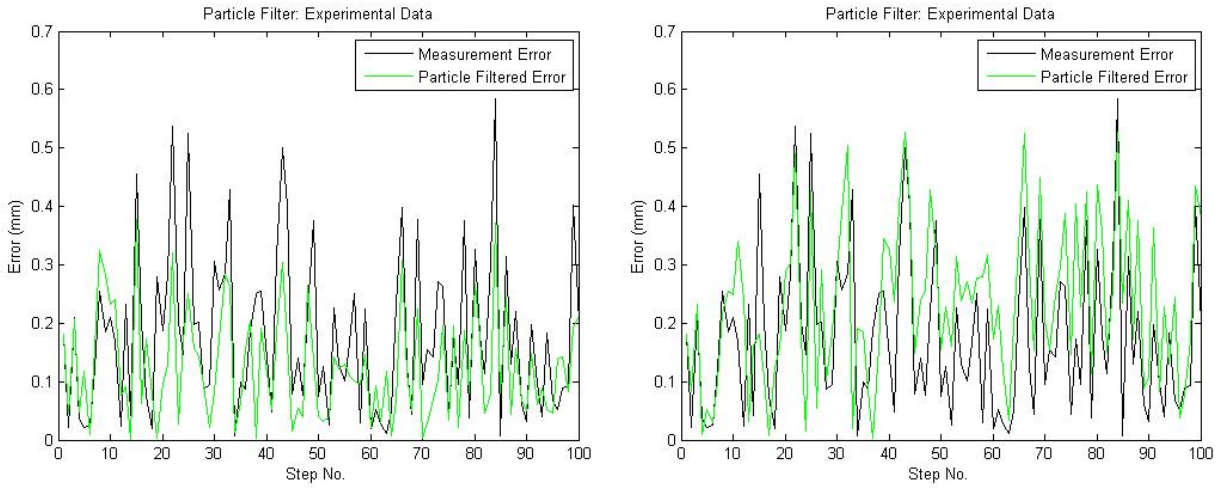
Figure 4.11: Simulated degeneracy - process noise, $\sigma = 0.0000001$.

number of particles that were approximating the state, even if not accurately. In extreme cases, all of the weights go to zero and a divide by zero warning is generated in Matlab when the weights are normalised - the sum of the weights is equal to zero.

The possibility of the random numbers generated just happening to lead the particles away from the true estimate is obviously not something that can be replicated on demand without purposely changing the parameters of the filter to lose particles to degeneracy. It has happened in the simulation when experimental values are used. This will be discussed presently.

Figure 4.12 shows the results of two runs of the simulation of the particle filter using the experimental data for one variable, (y-axis position, teat 1). The process noise standard deviation was set to a value of 1 and the measurement noise value was set to 0.5. This was higher than the actual value of 0.2 but it allowed the particles to diversify while still allowing them to converge. This simulation used the same system model as that used for the Kalman filter simulations. For the results shown in Figure 4.12a, the standard deviation of the measurements was brought down from 0.1339mm to 0.0911mm. Figure 4.12b shows another run of the same simulation. The performance is notably worse, with the resulting values having a standard deviation of 0.1336mm, an improvement of just

0.0003mm.



(a) (b)

Figure 4.12: Simulated results - experimental data. Plots (a) and (b) represent two runs of the same simulation.

The difference between the two runs of the simulation is accounted for by the random nature of the selection of the particles. This random nature of the filter can cause it to fixate on certain particles that may not be the best ones.

Particle deprivation is a major problem for the particle filter. As we have seen, certain parameters can have an effect on this. But even still, it is a problem inherent to the algorithm itself. The luck of the draw can result in there being no particles near the true state. If the sensor were to lose the target object momentarily as it moved, it would not be able to recover, since there would be no particles near the correct position of the object when it reappeared. In mobile robotics this is known as the kidnapping problem, where the robot is picked up and placed at another location on the map. This is mentioned [65] and in [66].

A possible way of solving the degeneracy problem is to add a certain amount of new random particles at the resampling stage of the algorithm. This will not depend on the values of the current states or on the distribution of the particle estimates. These will be particles distributed in state-space so that if the filter is tending to zone in on the wrong area, there will be particles introduced that will be closer to the true values. These will

be accordingly assigned higher weights and the mean of the particles will shift towards the correct values. A measure of degeneracy was put forward in [67]:

$$\hat{N}_{eff} = \frac{1}{\sum_{i=1}^{N} (w_k^i)^2} \tag{4.26}$$

where w_k^i are the normalised weights. The weights must be normalised so that their sum is 1. This is because they represent a discrete normal distribution.

Three methods of solving the degeneracy problem are compared in [65] one of which being the augmented Monte Carlo localisation algorithm discussed in [66] and which is found to outperform the other two methods. These other methods are sensor resetting localisation [68] and mixture localisation [69]. Although they refer specifically to localisation they are methods for adding particles at each iteration of the particle filter and can be used in any application.

The conclusion drawn from these simulations is that the particle filter is not suitable for use in this application due to a lack of reliability. The Kalman filter shows better improvements over measured values than the particle filter and is not prone to the degeneracy issues seen in the particle filter. While this could be improved, it would require further development to incorporate of the augmented Monte Carlo localisation method of adding particles.

4.6 Handling of Misidentification

4.6.1 Motivation

The work of [19] was concerned with the automated identification of the positions of the teats. A setup was developed and tested that used a thermal camera to detect the teats. Images from the thermal camera were then processed to identify the endpoint of the teats. A homography was applied to give a region of interest to a pair of stereo cameras allowing the positions of the detected teats to be identified and measured. It was found

that there could be several situations where the automated vision system being tested was unable to identify all four teat positions. These included situations where a teat was occluded by another and where a heat source behind the teats made one or more invisible to the thermal camera. Although the system was able to handle the situation where an unheated teat-shaped object was present it would be a lot harder to guarantee the absence of spurious heated objects in a milking parlour with other cows and machinery in the environment.

The teat identification problem is a substantial issue and is beyond the scope of the present work. It is thus assumed that the correspondence between teats in the original and new frame is known. In keeping with the idea of improving measurement values investigated in Sections 4.4 and 4.5, it was decided that techniques would be developed to allow successful placement of the milking cups even when an object other than one of the teats has been identified as a teat. Although techniques exist for robust identification, such as RANSAC [70] the problem posed here was deemed significantly more simple. While RANSAC is suited for use with clouds of points, the problem in this application uses only four points and it is expected that the majority of these will be inliers. An alternative solution was thus pursued.

It was proposed that accurate measurements could be taken of the positions of the teats of a cow either by verifying the measurements of the vision system or by using another more accurate measurement system (e.g. laser scanning). This would be performed just once for each cow when it is first introduced to the system. An RFID chip could identify a particular cow. This would likely be a feature of a fully automated milking parlour anyway so that information regarding the health of the cow, the quality and quantity of milk produced and biometric data could be recalled and updated. This biometric data could include the positions of the teats relative to each other. Using relative positions would mean that no matter where exactly the cow was standing in the stall, or its posture, the shape of the cluster would be known. Thus the position of a misidentified teat could be updated based on the known relative teat positions.

4.6.2 Detection of Misidentified Teat

In order to find which teat has been misidentified, the measured values are compared to the known values. For each of these sets of points a matrix of norms is found. These are the norms of the distances from each teat to every other teat. The difference between the matrices of norms for each set is found. Since these matrices are symmetric, one row and one column will show larger differences to the other rows or columns respectively corresponding to the i^{th} teat. The algorithm is presented in more detail below.

1. For the set of known teat end points, create four matrices containing four identical columns. In each matrix these columns will be equal to a different point, so that each matrix holds four copies of the same point.

2. Each of these matrices is subtracted from the set of known points resulting in four new matrices. Each of these contains as columns distances from the corresponding point to every other point.

3. The Euclidean norms of the columns of these matrices are found and combined into a matrix. This matrix is symmetric with zeros on the diagonal (i.e. the norms of the distances from the points to themselves).

4. Steps 1-3 are repeated for the set of measurements that includes the misidentified teat.

5. The difference between the two matrices of norms is found. The row (or column) with the largest sum represents the teat with the largest error.

This is represented mathematically in Equations (4.27) - (4.29). In these, p_i is a point vector corresponding to the i^{th} point in the matrix of definitive measurements. Likewise, m_i is a point from the matrix of measurements. $N_{i,k}$ is a 4×4 matrix of norms where i indicates the row and k indicates the column.

$$N^p_{i,k} = \|p_i - p_k\|_2 \tag{4.27}$$

$$N_{i,k}^{m} = \|m_i - m_k\|_2 \tag{4.28}$$

$$max_i = \left\| \sum_{i=1}^{4} |N_{i,k}^{m} - N_{i,k}^{p}| \right\|_{\infty} \tag{4.29}$$

This is very simple to implement in Matlab. This algorithm was found to correctly identify the worst point every time when an extra error was added to one of the four simulated points.

4.6.3 Orthogonalisation Algorithm

Once it is known that a teat has been incorrectly identified, it is possible to use historical knowledge of relative teat positions as well as measurement data to infer where the misidentified teat actually lies. This is done by projecting the historical data of the misidentified teat into the measurement frame using an orthogonal projection. Two techniques to calculate the orthogonal projection are developed and tested and are referred to as the Four-Point Orthogonalisation and the Three-Point Orthogonalisation respectively.

The line of inquiry that led to these algorithms can be summarised as follows. A set of points, ${}^{M}p$, is available that is known to have correct positions in the points own coordinate frame. A corresponding set of measured points is known in the vision system coordinate frame, ${}^{C}p$. Given the same set of points in two coordinate frames a transformation matrix between the two frames can be found.

$${}^{C}T_{M} = \frac{{}^{C}p}{{}^{M}p} \tag{4.30}$$

If one point in one of the coordinate frames has an error added to it, the two sets of points will not completely correspond and the resulting matrix will not be orthogonal. The set of teats including this additional error is referred to as $\widehat{{}^{C}p}$. The non-orthogonal

transformation matrix found using this set of points is given as:

$$\widehat{{}^{C}T_{M}} = \frac{\widehat{{}^{C}p}}{{}^{M}p} \tag{4.31}$$

Orthogonality is a condition of transformation matrices in Euclidean space. If this condition is not upheld then one can think of the mapping from one set of points to another set of unrelated points as warping space. This non-orthogonal matrix is then passed to an orthogonalisation algorithm (discussed below) and a new, orthogonal transformation matrix, $\widehat{{}^{C}T_{M_O}}$, is produced. This is then used to convert the original known measurements into the new coordinate frame.

$${}^{C}p_{O} = \widehat{{}^{C}T_{M_O}} *^{M} p \tag{4.32}$$

The point identified as the one with the largest error is then substituted from ${}^{C}p_{O}$ into the previous set of best estimates, $\widehat{{}^{C}p}$. These algorithms take a non-orthogonal transformation matrix and find the closest orthogonal matrix to it. The result will not be the same transformation as the ideal one (i.e. ${}^{C}T_{M}$) but it will be an approximation of it. This will allow the trusted points, M_p, to be mapped to corresponding points through an orthogonal matrix. The result is that the misidentified point with the largest error is closer to the true value.

A problem may arise in matching corresponding teats from the two sets. The teats in a set are always selected in the same order (see Figure 2.4). However it is conceivable that the misidentified teat will be identified as being in such a position relative to the other teats that this numbering system results in the teats being numbered incorrectly. In this case, the solution is to try to match each possible numbering configuration to the corresponding teats. All but the correct one will display large errors.

4.6.3.1 Four-Point Orthogonalisation

The four-point orthogonalisation procedure uses all four measured points, ${}^{C}p$, to construct the non-orthogonal transformation matrix $\widehat{{}^{C}T_{M}}$. This is then passed to an algorithm arrived at in consultation with Dr. Michael Clancy of the School of Mathematical Sciences at DCU. The derivation of this algorithm is presented in Appendix A. This is a variation of Cartan decomposition in Lie groups. This algorithm takes as its input a non-orthogonal n×n matrix, A, and finds the closest point, R, in the space of orthogonal matrices. This matrix, R, is essentially the closest orthogonal approximation of A. This is found using the following equation:

$$R = AS^{-1} \tag{4.33}$$

where,

$$S = P \begin{bmatrix} \sqrt{\lambda_1} & 0 & 0 & 0 \\ 0 & \sqrt{\lambda_2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sqrt{\lambda_n} \end{bmatrix} P^T \tag{4.34}$$

Here, λ_j, are the eigenvalues of $A^T A$ and P is the matrix of eigenvectors. The measured teats are transformed using the resulting matrix and the worst point in the set of measured points is replaced with its corresponding orthogonalised point.

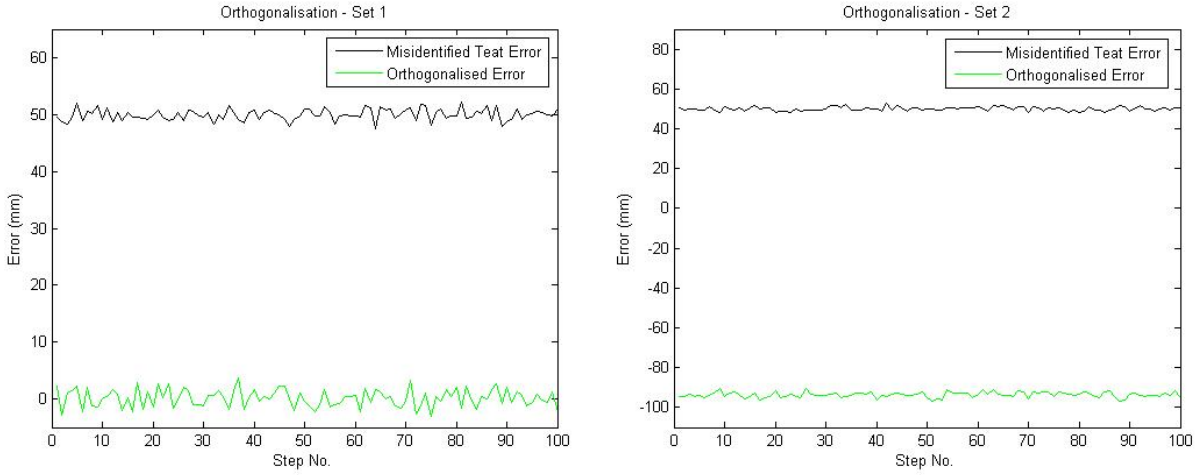
The four-point orthogonalisation algorithm was tested using measurements of the teat rig taken by the Microscribe. These were used as the definitive measurements, ${}^{M}p$. The set of points was then transformed into the camera coordinate frame using the camera to Microscribe transformation matrix, ${}^{C}T_{M}$. Simulated measurement noise was added to the x-, y and z-values of these points, ${}^{C}p$. An extra value was added to one of the teats in the form of a constant value on one coordinate (i.e. x, y or z). This was an arbitrary value set at 50mm which represented the error due a misidentified teat. This

value was chosen since it would produce an error outside of what would be likely to result from measurement noise. This value would also place the misidentified teat within the workspace of the end-effector so it could in theory be a teat. The detection routine described in Section 4.6.2 was then run and the problematic teat was detected.

The simulation then enters a loop in which the transformation matrix described in Equation (4.30) was found. This was then passed to the orthogonalisation algorithm and the result used to find the improved teat position. This loop iterates 100 times. Analysing the data returned (below) shows this to be more than sufficient to allow the algorithm to converge on a steady state transformation matrix, $\widehat{{}^{C}T_{M_O}}$.

The results of the simulation performed on two data sets are shown in Figure 4.13. Both of these data sets were from Microscribe measurements of the teat rig. The two sets represent the teats in two configurations that are likely to occur and can be measured by the vision system. In each case the test was repeated 100 times. Both data sets show that the teat identified as having the worst error is out by around 50mm. This is known to be the error the misidentified teat, so these plots both verify that the detection routine is working reliably. In the first plot, Figure 4.13a, the result of the orthogonalisation algorithm is consistently close to zero, exhibiting only measurement noise. The plot of the second data set, Figure 4.13b, however, shows that the algorithm consistently settles on the wrong result. Instead of converging on the correct set of coordinates for the point and the error going to zero (apart from measurement noise), the algorithm settles on another point, further from the true position than the misidentified point.

Some more insight can be gained by looking at the convergence of estimates within a simulation. Figure 4.14 shows how the errors of the coordinate values of the teat detected vary during the simulation when a 50mm error was initially added to y-axis coordinate. As before, the two plots show two sets of data - the same sets used in Figure 4.13. The plots demonstrate that the choice of 100 iterations for the internal loop is more than enough for convergence. Figure 4.14a shows that with the first data set, the point converges to a steady state within 10 iterations. In Figure 4.14b it takes far longer to converge, and as



(a) Data set 1.

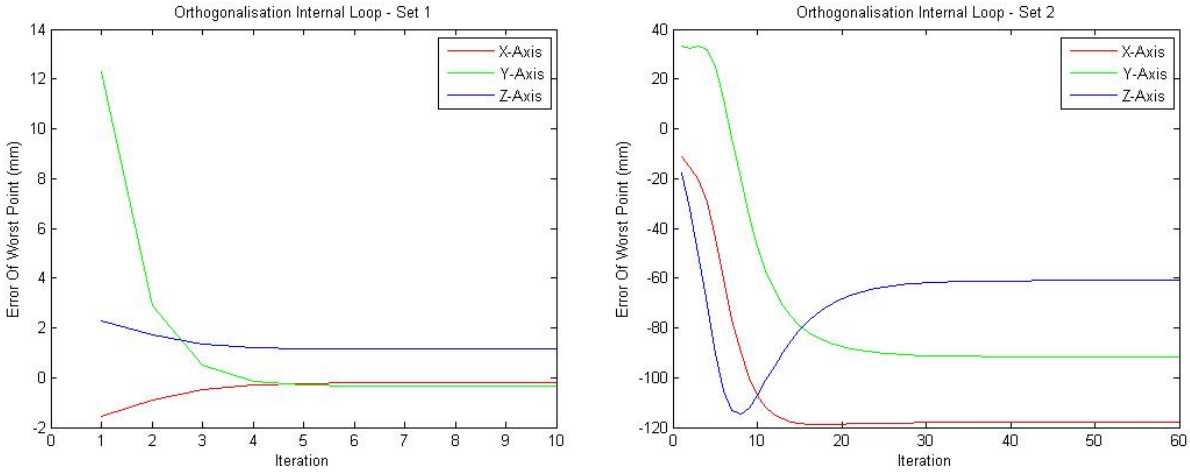
(b) Data set 2.

Figure 4.13: Orthogonalisation simulation results.

was seen in Figure 4.13, it converges on a point far from the true position. Comparing the two plots of the internal loop, we see that the algorithm applied to data set 1 exponentially approaches the correct values, but when applied to data set 2, each coordinate appears to converge on another point before changing direction and settling on the final position. This is most noticeable on the z-axis where the error value approaches -120mm, hitting a minimum value on the eighth iteration and eventually reaching a steady-state error value of about -57mm. The y-axis error value also changes direction although it is less noticeable and happens in the first 5 iterations of the internal loop. It is believed that the reason the algorithm changes direction is that the solution it finds is not unique. The algorithm was tested for a range of datasets and it would converge on the wrong point in approximately 25% of cases. The datasets were all measured from arbitrary teat configurations with no spatial characteristics connecting the sets that performed poorly.

4.6.3.2 Three-Point Orthogonalisation

In this algorithm, three points from from each set of measurements (^{C}p and ^{M}p) are used to create a transformation matrix. Since it is not possible to simply divide one set by the other to create the matrix using just three points in each, a coordinate frame is



(a) Data set 1.

(b) Data set 2.

Figure 4.14: Convergence of estimated coordinates.

created for each set of points using the three best teats. These coordinate frames can then be used to find an orthogonal transformation matrix. This algorithm does not require the iterative orthogonalisation seen in the four-point algorithm and is described in more detail presently.

The misidentified teat is detected as described in Section 4.6.2. The algorithm then calculates the mean, y_O, of the three other points in the set of vision system measurements, ${}^{C}p$. The vector from this centre point to the first point in the set of three points is found, y_x. This will be the x-axis of the new coordinate frame. Another vector, y_{int}, is found from the centre point, y_O, to the second point. The cross product of the two vectors is found. This produces a third vector perpendicular to the plane on which the other two vectors lie. This new vector will be the z-axis, y_z, of the new coordinate frame. The cross product of the x-axis and the z-axis is found to produce the y-axis, y_y. We now have three orthogonal axes that define the 3×3 rotation matrix of the transformation. With the centre point, y_O, as the origin this produces a complete transformation matrix from

the vision system coordinate frame to the measurements own frame:

$$ {}^{C}T_{y} = \begin{bmatrix} \begin{bmatrix} \begin{bmatrix} y_x & y_y & y_z \end{bmatrix}^T & \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \end{bmatrix}^T & y_O \end{bmatrix} \tag{4.35} $$

The same technique is used to create a transformation matrix from the coordinate frame used by the known measurements system (in testing this is the Microscribe frame, Section 2.1.5) to these measurements own frame. This frame is ${}^{M}T_{p}$.

Each of these transformations is used to convert their respective set of measurements into corresponding points, such that:

$$ \widehat{{}^{y}p} = ({}^{C}T_{y})^{-1} * \widehat{{}^{C}p} \tag{4.36} $$

and

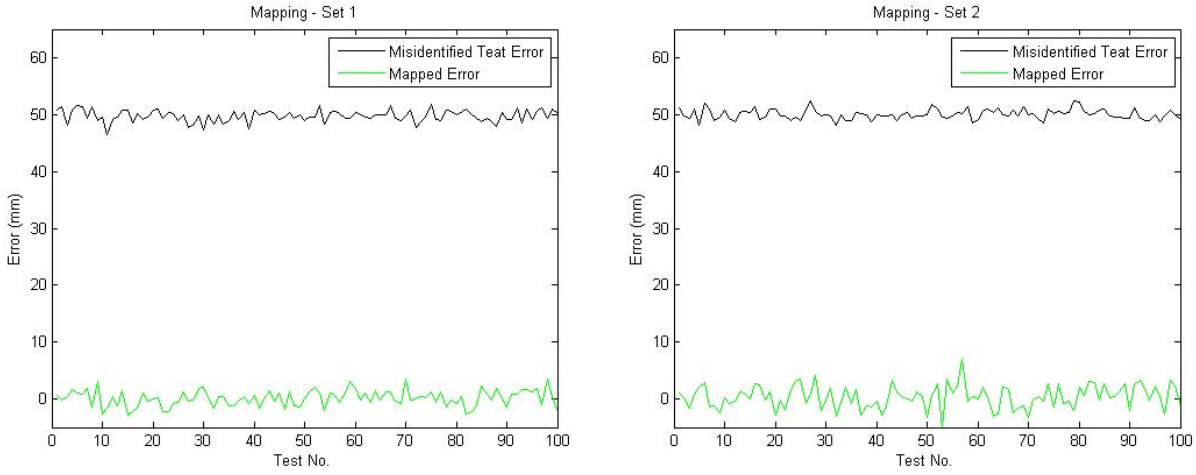
$$ {}^{p}p = ({}^{M}T_{p})^{-1} *^{M} p \tag{4.37} $$

As in previous sections, $\widehat{{}^{C}p}$ is the set of measurements that includes the additional misidentification error. The two sets of points resulting from Equations (4.36) and (4.37) both represent the same set of teats, defined in their own coordinate frames and should therefore give almost identical points apart from the misidentified point in $\widehat{{}^{y}p}$. This detected misidentified point is then replaced in $\widehat{{}^{y}p}$, with its equivalent taken from ${}^{p}p$.

Since this algorithm is directly replacing the worst measured teat with its definitively measured counterpart, it is not necessary to iterate any part of the process.

The implementation of this algorithm was a lot more straightforward than the orthogonalisation algorithm. It was also a lot more computationally efficient since it did not need to be looped. The algorithm took a lot less time to run as a result. The known measurements, stereo vision system measurements and the additional misidentification error were found the same way as described in Section 4.6.3.1. Figure 4.15 shows the

results of tests using the same two sets of data presented in Figure 4.13. Again, the test was run 100 times for each set. The algorithm performs well for both sets, and indeed for any other set of points it was applied to. The simulations show the mapping routine to be more reliable than the orthogonalisation algorithm. One thing worth noting about this however is that the standard deviation of the error values is larger for the results than for the raw measurements. The measurements have a standard deviation of about 0.896mm which goes up to about 2.088mm after the mapping algorithm. The reason for this is because the measurement noise error is present when the transformation matrices are found and although they are orthogonal, the measurement noise present causes them to be slightly different to the true values.



(a) Data set 1.

(b) Data set 2.

Figure 4.15: Mapping algorithm simulation results.

Chapter 5

Testing & Results

5.1 Actuation System Component Accuracy

To ascertain the accuracy of the individual motors in the end-effector a series of movements was made with each motor and the resulting changes in position were measured using the microscribe. The four linear actuators (stepper motors) were moved from their home positions ten times in steps of 15mm. The resulting measurements are summarised in Table 5.1. The values presented are the means and standard deviations of the move error. The move error is the difference between the commanded and actual move distance.

Table 5.1: Mean and standard deviations of errors on the four linear actuators.

	Linear 1	Linear 2	Linear 3	Linear 4
Mean Error (mm)	-0.083	0.001	0.035	0.513
Standard Deviation, σ (mm)	0.126	0.068	0.113	0.239

The four revolute axes (servo motors) were moved through a full 360° in steps of 36°. These measurements are summarised in Table 5.2. This data is given in degrees rather than millimetres since it represents rotational errors. When measuring the positions of the revolute axes, the slack was taken out of the gears in the same direction each time. This effectively removed the effects of backlash from the measurements. The backlash in the servomotors was also measured separately. The angle of the backlash in each servo is

given in Table 5.3. There is no appreciable backlash in the stepper motors on the linear axis due to the motor being held at the current step when powered.

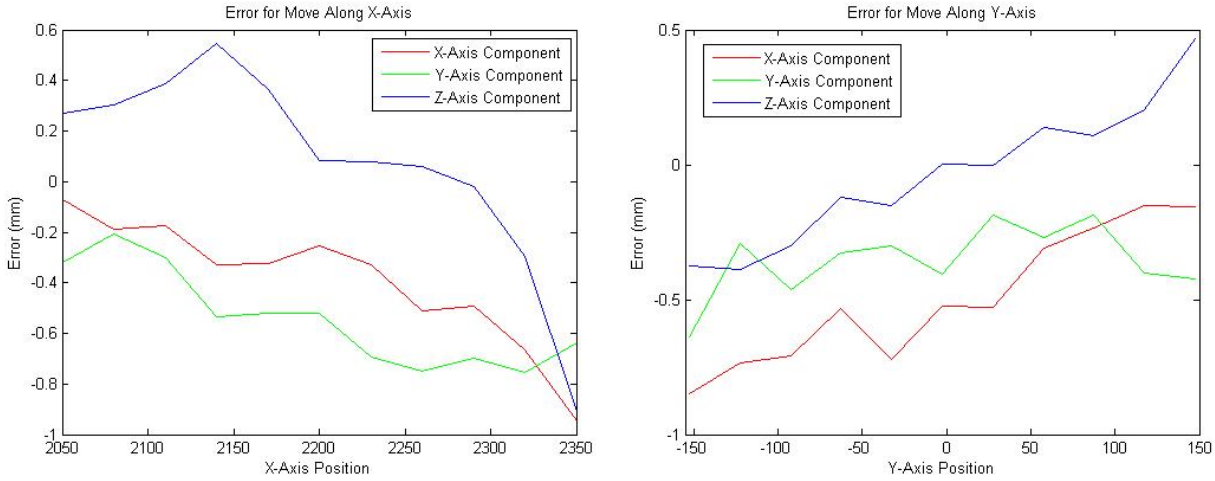
Table 5.2: Mean and standard deviations of errors on the four revolute actuators.

	Revolute 1	Revolute 2	Revolute 3	Revolute 4
Mean Error (°)	0.282	-0.825	-3.403	-0.216
Standard Deviation, σ (°)	1.051	1.133	2.568	0.18

Table 5.3: Backlash measured on the four revolute actuators.

	Revolute 1	Revolute 2	Revolute 3	Revolute 4
Backlash (°)	1.599	1.550	1.431	1.748

A series of measurements of movements of the 6-axis robot itself was taken. The toolplate of the robot was not able to reach the working area of the microscribe and the vision system. A point on the end-effector with a constant offset position from the toolplate was therefore measured instead. The robot was moved to a series of points lying parallel to the robot coordinate frame's x-axis and the positions measured. This was then repeated using a series of points lying parallel to the robot frame's y-axis. These results are graphed in Figure 5.1 and summarised in Table 5.4.



(a) X-direction

(b) Y-direction

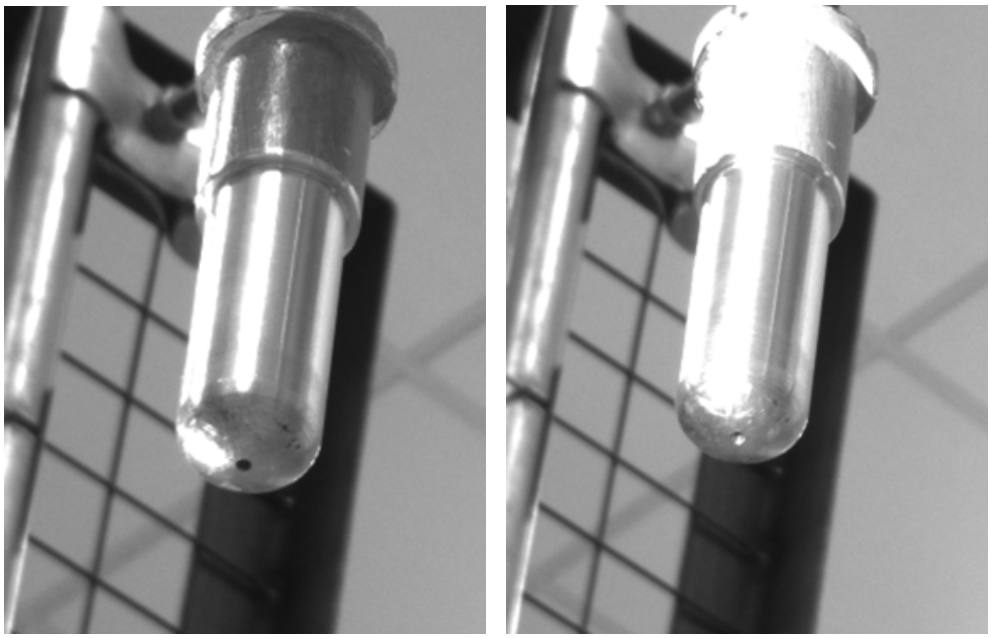
Figure 5.1: Positioning errors of 6-axis robot

Table 5.4: Mean and standard deviations of robot positioning errors in x- and y-directions.

	x	y	z
Mean Error (mm)	-0.44	-0.45	0.02
Standard Deviation, σ (mm)	0.25	0.186	0.34

5.2 Stereo Vision Pixel Level Accuracy

In the laboratory setup, the endpoint of each aluminium dummy teat was marked with a small hole drilled axially into the end of the teat. This hole measures approximately 2mm across. The light source was positioned so that the hole would show up as a dark region on the illuminated aluminium background, see Figure 5.2a. To identify a teat in the image, the centre of this area is selected by clicking with the mouse in the Labview VI.



(a) Suitable lighting. (b) Poor lighting.

Figure 5.2: One of the dummy teats as seen by one of the stereo camera pair.

Selecting a point on the image by hand naturally includes a degree of variance, for several reasons. Since we are selecting the centre of an area, the best pixel is not always obvious and can come down to the user's judgement. It is also not possible for a user to

reliably select the same pixel twice using the mouse. This is because the Labview image object does not allow for the image to be zoomed or panned while the VI is running. As a result, when selecting the teats, it is not possible to zoom in on an individual teat's endpoint for more accurate identification. Lighting conditions can also have an impact and although it was attempted to maintain consistent lighting, the fact remains that depending on the position of the light source and indeed the weather conditions, the hole marking the endpoint could at times appear larger or smaller or misshapen due to a glint of light on the edge of the hole, Figure 5.2b.

It was found that an error in the 3D coordinates generated by the stereo triangulation of up to several millimetres could occur depending on what pixel on the teat was selected. It was thus decided to measure the difference in the resulting point if an adjacent pixel was instead selected. The checkerboard was positioned in front of the stereo cameras at approximately the usual working distance. A series of corners on the board, approximately along a horizontal line passing throught the centre of the view of the vision system, were measured. Figure 5.3 shows a representation of these points indicated by red numbers.

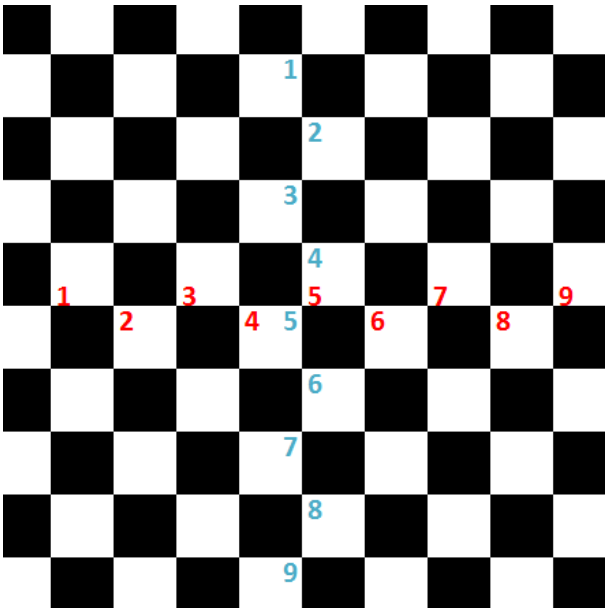
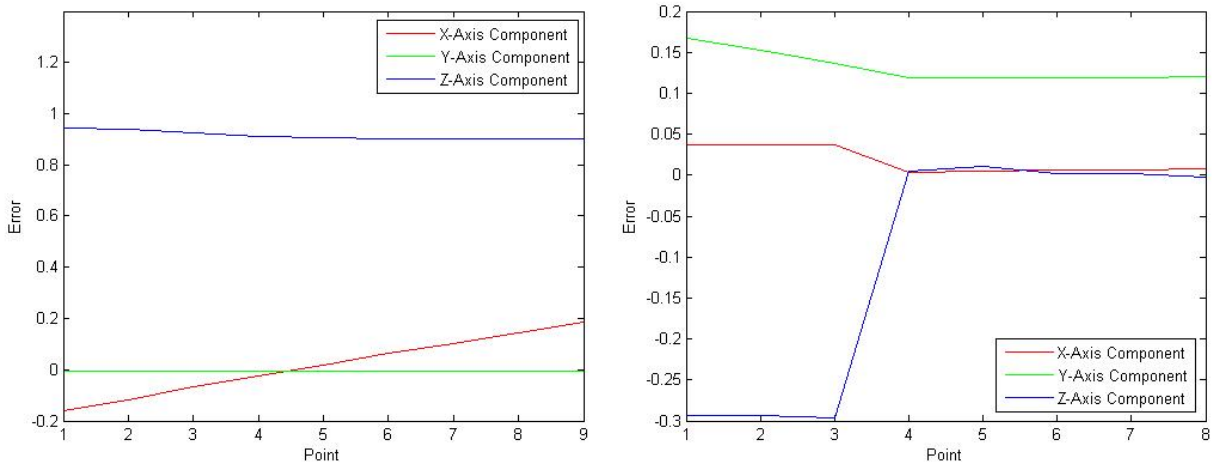


Figure 5.3: Test points (corners) taken to measure pixel level accuracy across width of view (indicated by red numbers) and along the height (blue numbers).

This was repeated for a vertical line, again as close to the centre in both images as possible (corners indicated by blue numbers in Figure 5.3). The triangulation algorithm returned 3D coordinates for these points and the pixel values in both images were recorded. These pixel values were then modified such that for the horizontal line of points, the x-direction pixel value on the right hand image was shifted by one pixel. For the vertical line of points the y-direction pixel value on the right hand image was shifted by one pixel. These new pixel coordinate were then input into the triangulation algorithm and new 3D coordinates were produced. These new coordinates represented a point that was exactly one pixel away from its corresponding point in the original set of points. This gives a measure of the difference a pixel makes around the working area of the camera system. The errors of the chosen points are shown in Figure 5.4 where Figure 5.4a represents the points chosen along the x-axis that are shifted one pixel on that axis and Figure 5.4b shows the errors of points chosen along the y-axis again shifted one pixel on that axis.



(a) X-direction

(b) Y-direction

Figure 5.4: Pixel level errors of the vision system.

5.3 Actuation and Sensing

5.3.1 Motivation

In tests using the milking vacuum cups, it was found that the cups could be placed on the teats every time. This level of accuracy in the combined vision and actuation systems met the most basic of criteria, the ability to place the four milking cups onto the teats. However, although this was seen as a major success, it was not easily quantifiable and did not indicate any systematic errors in the system. Such errors could combine with other unforeseen problems in the future to decrease the accuracy to the point that the system would become incapable of reliably placing the cups on the teats. Thus, it was necessary to have a good understanding of the nature of the errors of the system. This was done by measuring the positional accuracy of the system (using the Microscribe) and measuring the sensing accuracy.

In order to assess the accuracy of measurements by the stereo vision system, that incorporate the transformations used (i.e. converting coordinates from the camera system's coordinate frame to the coordinate frame of the robot) a series of tests were performed. These tests were designed to also provide information about how accurately the robot was capable of positioning the milking cups.

5.3.2 Test Description

The dummy teats were set to an arbitrary configuration such that all four teats were within the field of view of the cameras and were accessible by the end-effector. The teats were measured by the vision system six times. This figure was chosen to correspond to later tests where the vision system is mounted to the end-effector (see next section). The robot was then sent to the final measured position. Instead of bringing the milking cups to the teats, for testing, the revolute arms of the end-effector were replaced with simple beams of aluminium. These beams were marked at the position where the centre of the milking cups would be found, i.e. the distance from the axis of rotation to the

mark was set equal to the distance from the axis of rotation to the central axis of the milking cup. This allowed more accurate measurement of the final position of the robot. It would not have been practical to use the actual milking cups since the central axis passes through the opening of the cup. The final positions reached by the end-effector were measured using the microscribe. The positions of the teats were also measured using the microscribe. During this phase of testing the cameras were in a fixed position on the floor close to the teat rig. The test was repeated for twenty different teat configurations. Six measurements of four teats in twenty configurations resulted in 480 data points to assess the accuracy of the of the stereo vision system. The test also resulted in 80 data points providing information about the accuracy of the hardware.

The results are generally presented in the world (robot) coordinate frame, however, for the sake of measuring the errors specifically associated with the vision system, the cameras' own frame will be used. This means converting points measured by the microscribe from the microscribe frame to the camera frame. This transformation is referred to as ${}^{C}T_{M}$ and is defined as:

$$ {}^{C}T_{M} = {}^{R}T_{C}^{-1} * {}^{R}T_{M} \tag{5.1} $$

where ${}^{R}T_{C}$ is the transformation from the robot to camera frame or can be considered as the description of the camera frame in robot frame coordinates. ${}^{R}T_{M}$ is the transformation from the robot frame to the microscribe frame. The procedures for setting up the transformations ${}^{R}T_{C}$ and ${}^{R}T_{M}$ are described in Sections 3.2.4 and 2.1.5 respectively. The use of the transformation matrix ${}^{C}T_{M}$ does not exclude any errors that may be associated with the transformations between the camera, robot and microscribe coordinate frames but it does let us see the correlation between measurement errors and the position of the test point on each axis of the vision system.

The measured positioning errors on the x-axis of teat 1 are graphed in Figure 5.5. This is typical of the errors seen on all axes of the four teats. The errors on all axes and

all teats are summarised in Table 5.5.

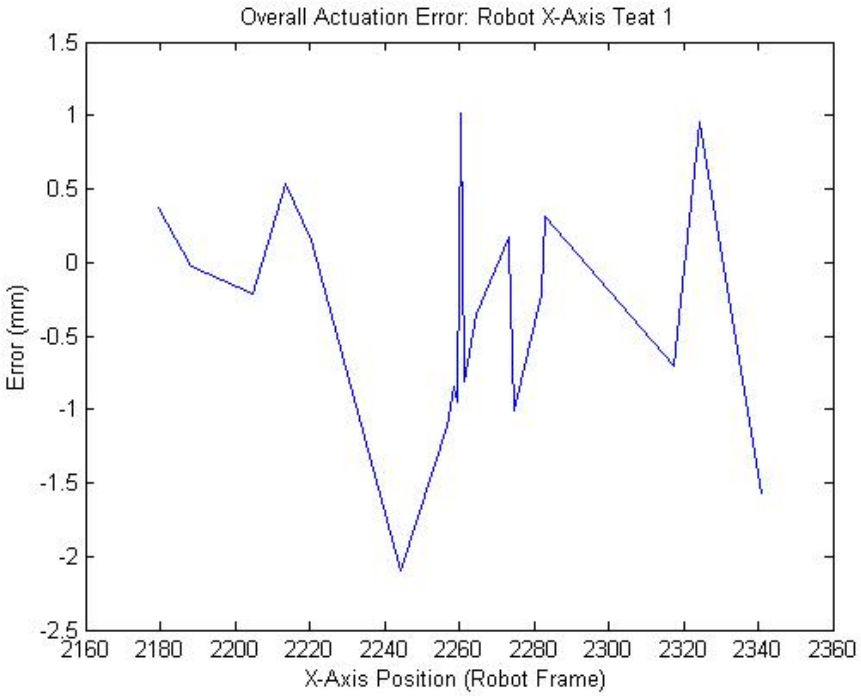


Figure 5.5: Positioning errors on x-axis of teat 1.

The same set of tests also yielded data pertaining to the accuracy of the vision system. This is summarised in Table 5.6. This data is also presented graphically in Figure 5.6. This shows the errors in the order the measurements were taken. Figure 5.7 shows the same data, this time with the errors plotted against the value of the measurement. This allows us to correlate the errors with spatial position.

5.4 Moving Vision System

5.4.1 Motivation

Mounting the cameras on the ground close to the teat rig provides an adequate view of the working area in the lab. An alternative option was to mount the cameras on the end-effector. The advantage of this is that no matter where the cow is in the stall (or where the cluster of dummy teats is in the rig) it will be possible to position the cameras

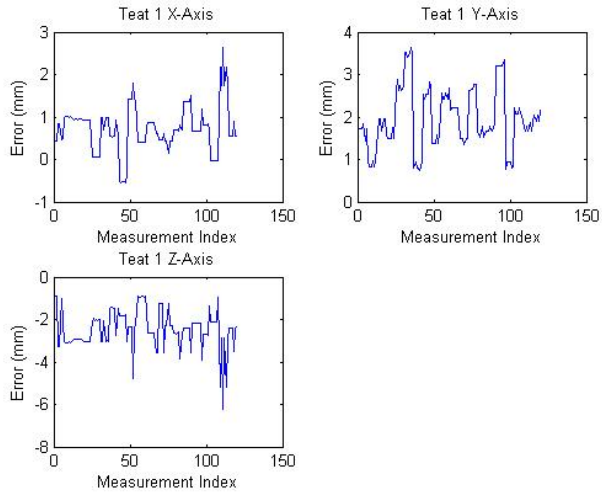
Table 5.5: Mean and standard deviations of positioning errors.

Teat	1		2	
Axis	x	y	x	y
Mean (mm)	-0.06	-1.63	-1.94	-1.03
Standard Deviation, σ (mm)	0.857	0.712	1.235	0.774
2σ (mm)	1.714	1.424	2.47	1.548

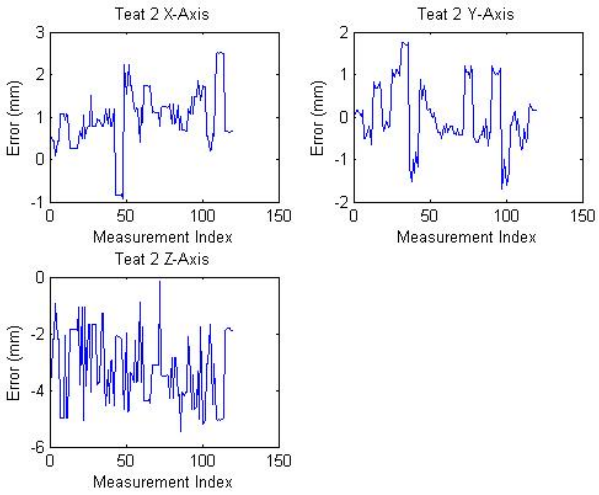
Teat	3		4	
Axis	x	y	x	y
Mean (mm)	-0.62	-2.45	-0.75	-3.44
Standard Deviation, σ (mm)	0.756	0.843	1.219	1.175
2σ (mm)	1.512	1.686	2.438	2.35

Table 5.6: Mean and standard deviations of vision system measurement errors in camera coordinates.

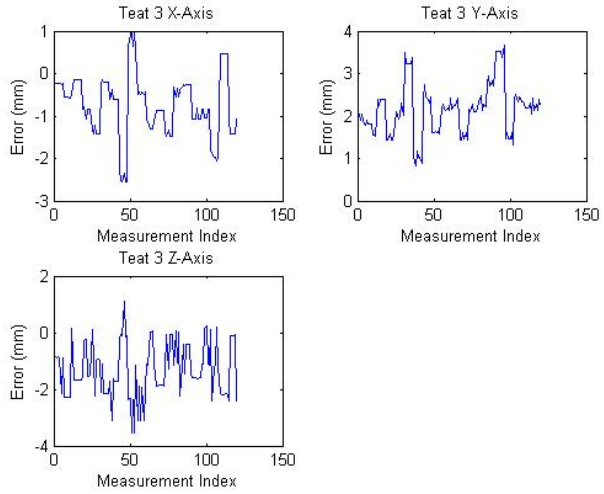
	x	y	z
Mean, (mm)	-0.09	1.46	-2.89
Standard Deviation, σ (mm)	0.988	1.15	1.171



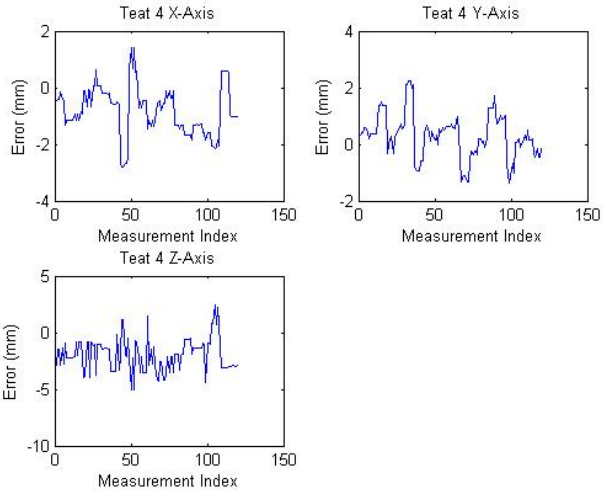
(a) Teat 1



(b) Teat 2

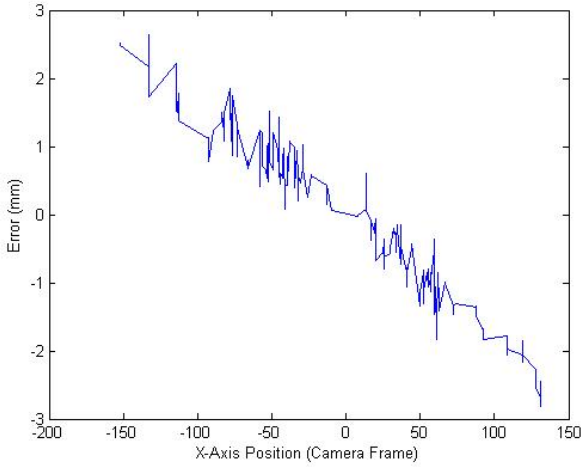


(c) Teat 3

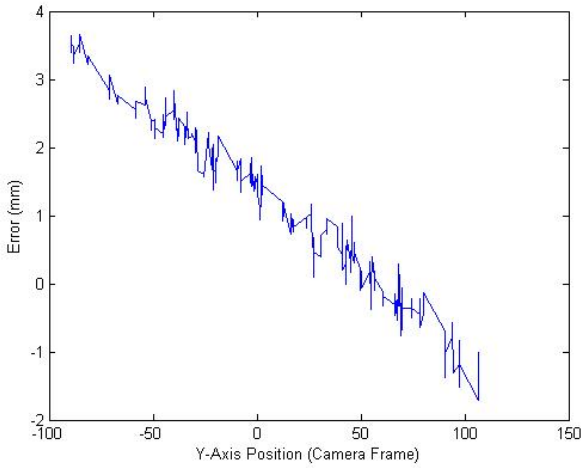


(d) Teat 4

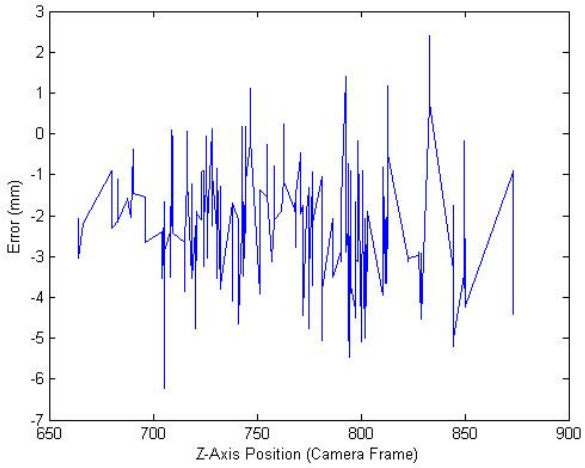
Figure 5.6: Vision system measurement errors plotted in order measurements were taken. Plots in groups of three (one for each axis).



(a) X-axis



(b) Y-axis



(c) Z-axis

Figure 5.7: Vision system measurement errors plotted against position. Errors shown for all teats, three axes.

in the best location to view the teats. The robot will be able to keep the teat cluster in the centre of the view. Although the teat rig in the lab provides an unobstructed view of the teats, in a working dairy parlour, it would be considerably more difficult to find one single position for the cameras where the teats were unobstructed by objects in the parlour (e.g. the kick rail of the stall, the legs of the cow) and without the vision system itself obstructing the robot. It was proposed that the cameras take multiple measurements as the robot approaches and use these measurements to home in on the teats.

Testing was thus performed to measure the accuracy of the cameras mounted to the end-effector as the robot approached with the aim of developing the system further to incorporate a more advanced tracking system. The goal of this test was to ascertain the errors of the camera measurements over a range of distances far greater than the range covered by the teats. The cameras would be measuring the teats over a range of distances from 1200mm down to 400mm. When the cameras are mounted on the floor the depth of view required is around 650mm up to about 875mm. By contrast, over the distances seen in this test, the teats will initially be out of focus and the error is expected to start large and decrease to levels seen in the previous test as the cameras get closer. This effect could be reduced by using a smaller aperture size to give a larger depth of field, however this will have the effect of making the image darker. While this could be remedied by increasing the gain settings of the cameras, the increased noise levels would be undesireable.

5.4.2 Test Description

The cameras were mounted to the underside of the end-effector in such a way as to provide a clear view of the teats without obstructing the mechanisms of the end-effector. As in the previous test, the teats were moved to arbitrary positions. These positions were measured using the microscribe. The robot was retracted to an initial position and moved toward the teat cluster in six stages. Six measurements provided sufficient data

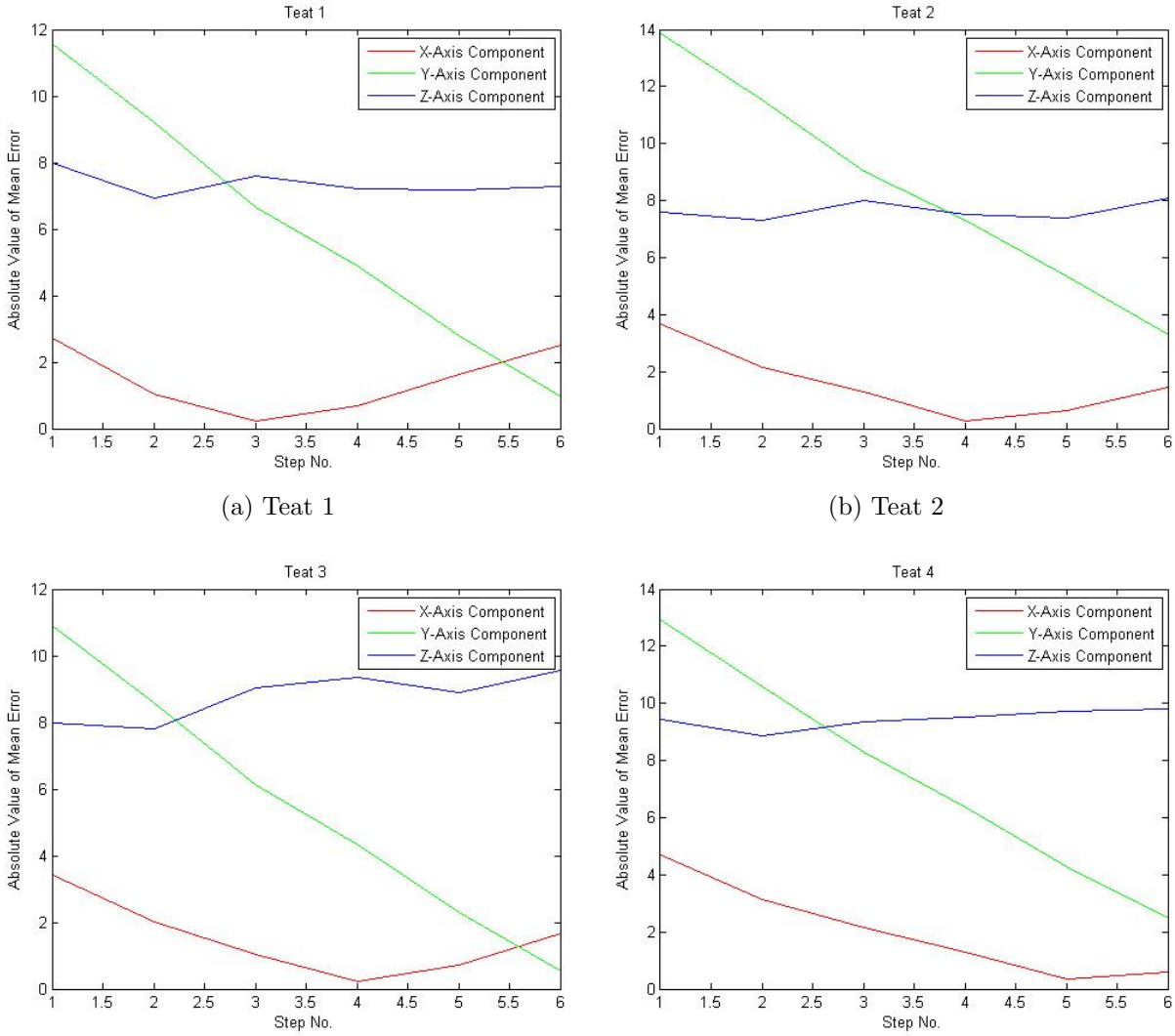
to analyse the change in measurements as the vision system approached without each test becoming excessively time consuming. At each step, a set of measurements was taken using the stereo vision system. The robot moved the end-effector into the positions measured at the last step. For each set of teat positions the vision system would acquire six sets of measurements each taken at a different distance from the cluster. This would again provide 480 data points for the camera measurements.

The results of this set of tests are shown in Figure 5.8. The error values have been averaged for each step of the approach, therefore, step 1 on the horizontal axis is the mean of the errors at step 1 of all tests.

5.5 Kalman Filtering

5.5.1 Motivation

The Kalman filter developed in simulation was tested on the hardware in the lab. This allowed the practical application of the filter to be evaluated for suitability as an integral part of the system. In this test, the cameras are mounted on the floor i.e. they are stationary and track the teats as they are moved at a constant velocity. This allows us to track points moving relative to the camera coordinate frame without complicating the movement by having the cameras moving too. This is not the same as measuring stationary teats with a moving vision system as in that situation the relative movement is known. Here, the movement of the teats is not known in advance. Indeed the main motivation of this test is to account for movement in the udder. This also removes any complications due to cameras being out of focus at greater distances and problems arising from the changing transformation matrix, ${}^{R}T_{C}$.



(a) Teat 1

(b) Teat 2

(c) Teat 3

(d) Teat 4

Figure 5.8: Absolute mean measurement errors. Each step is the mean of values from twenty tests.

5.5.2 Test Description

The teat rig was modified to allow the four teats to move together along a fixed path. This was achieved by means of a graduated slider, accurate to 1mm. The path of the movement lay approximately in the direction of the robot y-axis. The slider was attached between the frame of the teat rig and the teat mounting plate. This meant that when the slider was adjusted, all four teats moved together and did not move relative to one another. This assembly is shown in Figure 5.9.

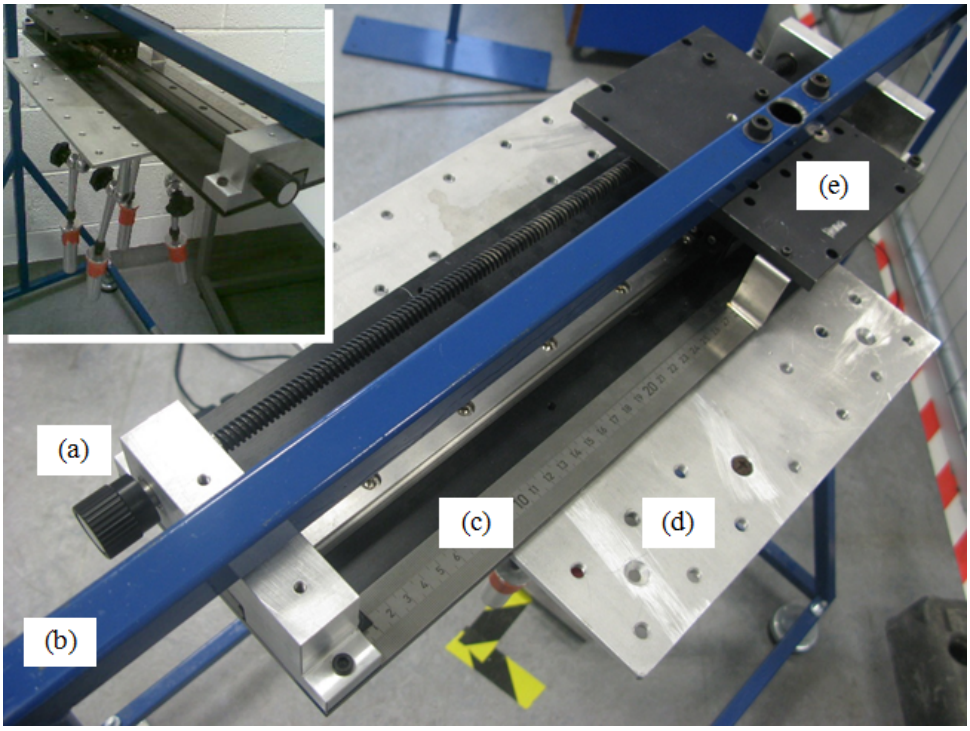


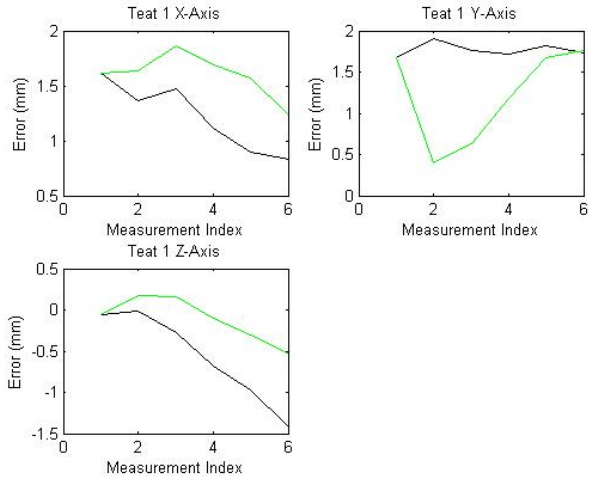
Figure 5.9: Sliding assembly (a) Knob adjusted screw. (b) Teat rig frame. (c) Graduated scale. (d) Teat mounting plate. (e) Sliding mechanism mounting plate. Inset shows the teats beneath the mounting plate.

The Kalman filter developed in simulation was modified to allow it to be run as part of the control system in Labview via a Mathscript node. Since the simulated system was designed to imitate the actual system and was found to run quite well, the covariance matrices used in the simulation as well as the system model were used in the application of the Kalman filter.

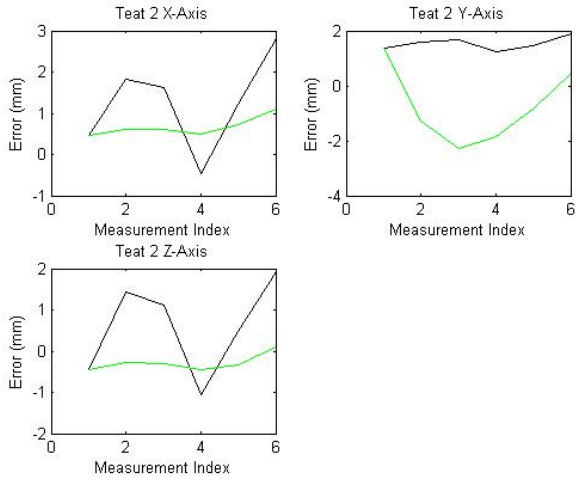
The first stage of the test comprised 20 runs. In each run, the teat cluster was moved 5 times by an amount of 3mm, giving 6 sets of points. The results from one such run are shown in Figure 5.10. These are typical of the entire data set and are discussed in Section 6.5.1.

Based on the outcome of this it was decided that another stage of testing was required involving a longer run of 20 movements, each again of 3mm, giving 21 sets of points. This was due to issues with settling time.

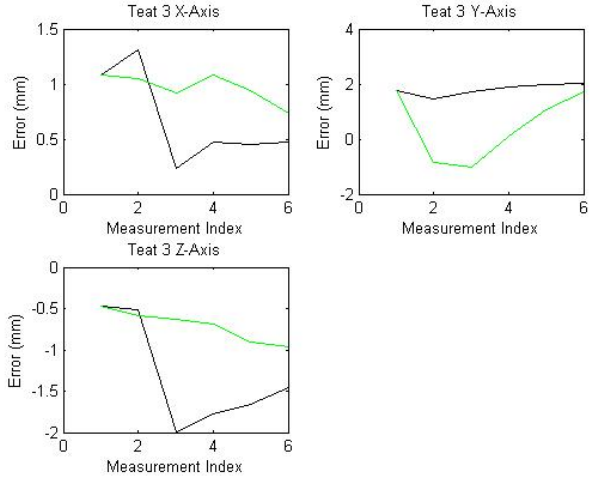
Taking so many more points took the test a lot longer to perform. Repeat tests, although entirely possible, would be slow. It was decided to save all relevant data (left



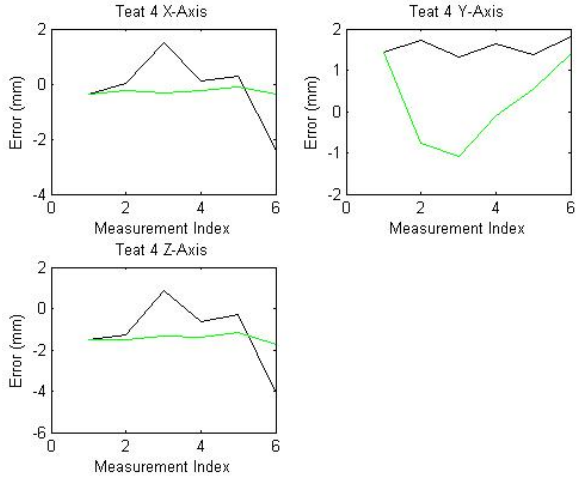
(a) Teat 1



(b) Teat 2



(c) Teat 3



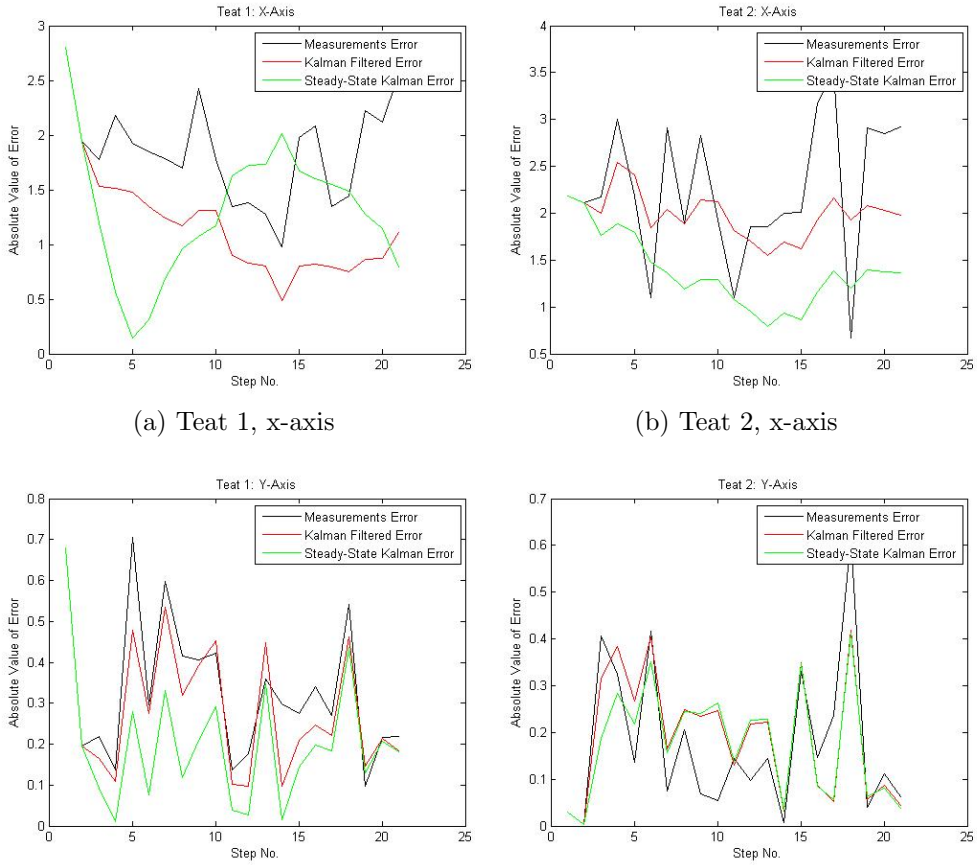
(d) Teat 4

Figure 5.10: Results from one six-iteration run of the system incorporating the Kalman filter.

and right image pixel values, vision system measurements, Microscribe measurements) and to use it as an input to the Kalman filter in a Matlab script, similar to how the simulations were run. Unlike the simulations however, these scripts used real-world data as opposed to simulated data and produced the exact same results as would be produced by the script in Labview. The advantage is that the test can be run multiple times using identical measurement values.

The test data was filtered using two methods, with the first, using the usual time-variant Kalman filter and with the second, a steady-state Kalman filter which used the

final Kalman gains from the first method. The Kalman gain update equation was left out of this test so the gains were not re-calculated as usual on this second pass. This is discussed in greater depth in Section 6.5.2. Figure 5.11 shows the outcome of this test for the x- and y-coordinates for two of the teats.



(a) Teat 1, x-axis

(b) Teat 2, x-axis

(c) Teat 1, y-axis

(d) Teat 2, y-axis

Figure 5.11: Results of both the time-variant and steady-state Kalman filter performed on the same experimental data set.

Following on from the results of this stage of the test, another test was performed in which 100 movements were used, each of 1mm. The smaller movement was necessary in order to keep the teat cluster within the field of view of the vision system. Using such a large number of iterations means that any issues with settling time will have a reduced impact towards the end of the test. The data from the 100 iterations was also input to the time-variant Kalman filter.

Several modifications were made to the Kalman filter. These are summarised in Table

5.7 The filter was tested at each stage using the same measurements from the 100 move run. The results are summarised in Table 5.8 where the standard deviation of the resulting error on each axis of each teat coordinate are presented. To represent this graphically, the results for one axis of one teat are shown in Figure 5.12.

Table 5.7: Summary of Kalman filters tested. cov() = covariance found using Matlab cov() function. x_M = Microscribe measurements. x_{sim} = simulated positions. $x_M(0)$ = Microscribe measurements at the origin.

Kalman Filter	R	Note
(i)	$(B\sigma)(B\sigma)^T$	Figure 5.12a
(ii)	$\text{cov}(x_M(0))$	Figure 5.12b
(iii)	$\text{cov}(x_M - x_{sim})$	Figure 5.12c

5.6 Handling of Misidentified Teats

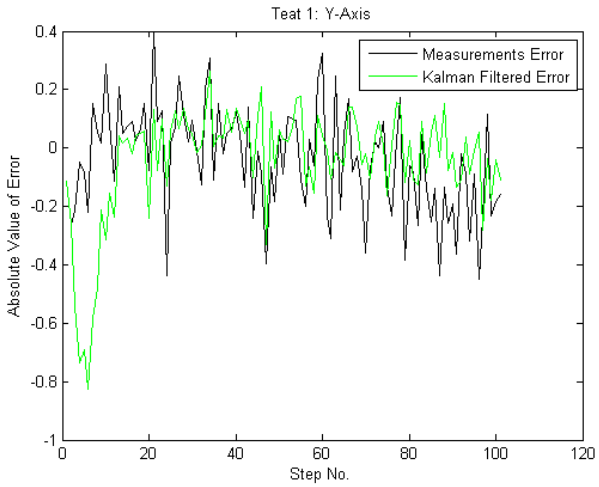
5.6.1 Motivation

In the cluttered environment of a milking parlour it is possible that an object other than the teat of a cow will be identified as a teat. In this situation the position of the misidentified teat will be quite far away from the teat cluster and will result in the system being unable to apply the milking cup. Although the teat identification system has not yet been developed, it is expected that misidentification could be a problem. The proposed solution to this problem is to take definitive measurements of the teats of a particular cow either through manually checking that the teats have been identified or through the use of another measurement technique (e.g. laser scanner). From this information a coordinate frame is set up based on the locations of the teats and these locations are defined in this coordinate frame. This will give the positions relative to one another and independently of the position of the animal in the stall. These relative positions will then be used to find the expected position of the teat that has been misidentified. Two algorithms were developed capable of this and are described in Section 4.6.

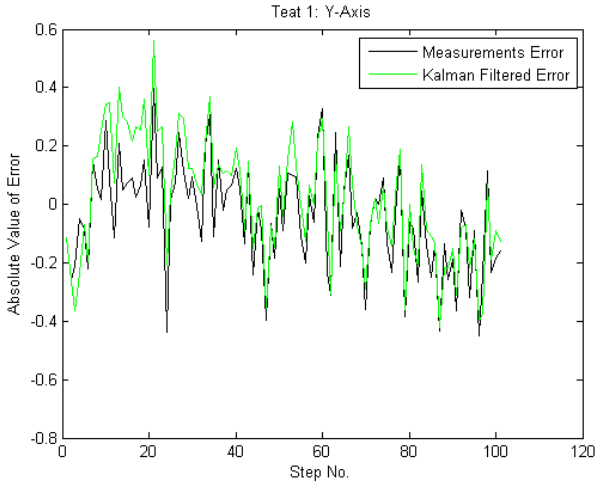
Table 5.8: Standard deviations of errors of measurements and 4 sets of Kalman filter estimates.

Teat	1			2		
Axis	x	y	z	x	y	z
Measurements Error Standard Deviation, σ (mm)	0.77	0.17	0.83	0.77	0.24	0.74
Kalman Error (i) Standard Deviation, σ (mm)	0.34	0.1	0.23	0.1	0.1	0.1
Kalman Error (ii) Standard Deviation, σ (mm)	0.36	0.16	0.24	0.19	0.17	0.19
Kalman Error (iii) Standard Deviation, σ (mm)	0.47	0.15	0.4	0.41	0.2	0.38

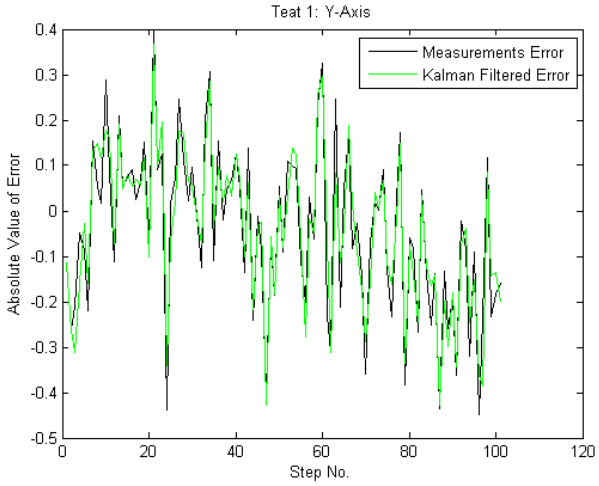
Teat	3			4		
Axis	x	y	z	x	y	z
Measurements Error Standard Deviation, σ (mm)	0.79	0.19	0.76	0.7	0.17	0.67
Kalman Error (i) Standard Deviation, σ (mm)	0.12	0.14	0.12	0.12	0.11	0.1
Kalman Error (ii) Standard Deviation, σ (mm)	0.19	0.18	0.17	0.22	0.15	0.18
Kalman Error (iii) Standard Deviation, σ (mm)	0.35	0.16	0.31	0.47	0.16	0.39



(a) Kalman configuration (i)



(b) Kalman configuration (ii)



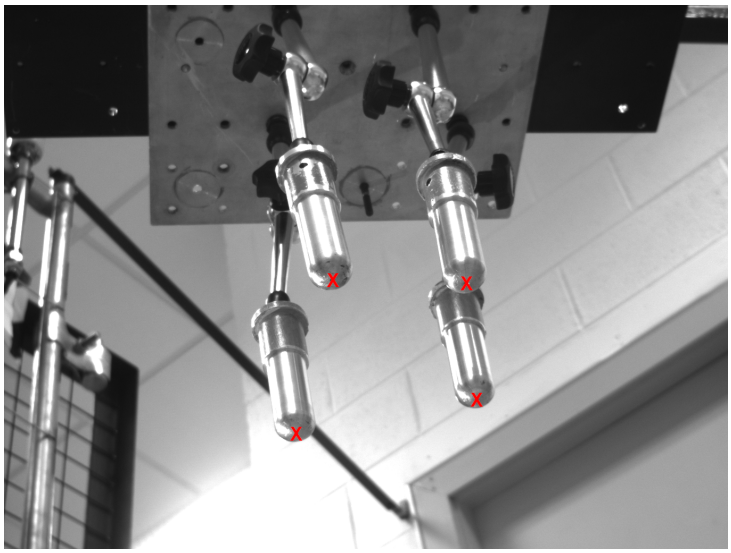
(c) Kalman configuration (iii)

Figure 5.12: Measurement and Kalman estimate errors for 100 move data. Results shown for teat 1, y-axis.

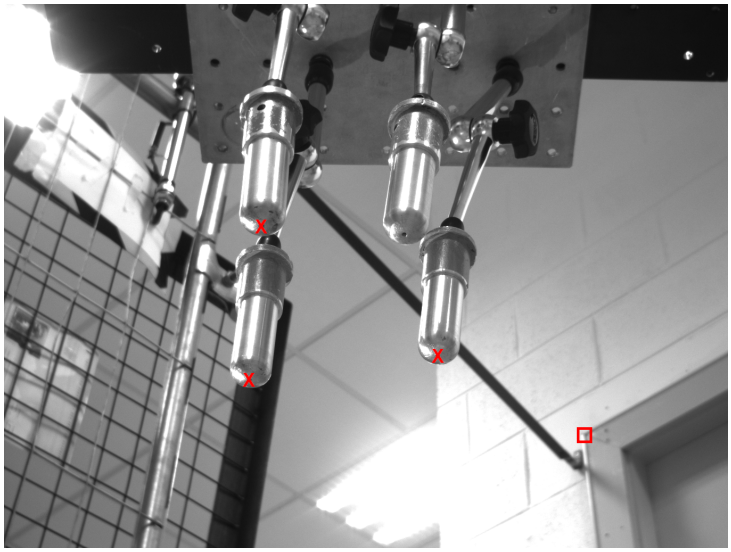
5.6.2 Test Description

The two algorithms were tested by measuring the positions of the teats with the cameras mounted on the floor. The teats were kept stationary and their positions were measured using the Microscribe and the vision system. When measuring with the vision system, the four teats were selected correctly in the left image but only three were selected in the right image. A point in the background of this image (the corner of a doorframe) was selected instead of the teat. This was repeated for twenty arbitrary sets of teat positions. For each repetition of the test a different teat was chosen as the one to be misidentified. The teat to be misidentified was cycled through the four teats in turn. In Figure 5.13, the selected teats are marked with a red 'X'. In the right-hand image (from the right camera) teat 3 has not been selected. Instead a background point, marked with a red square has been chosen.

The size of the error generated using this technique depends on the teat chosen, since some teats are closer in the right-hand image to the selected point than others. For example, referring to Figure 5.13 we can see that the point marked with the square is closer to 4 and thus causes a smaller error when that teat is misidentified. Similarly, teat 1 would result in a far larger error. The measurement error introduced in each test is shown in Figure 5.14 as are the results of the two algorithms for a set of 20 tests.



(a) Left.



(b) Right.

Figure 5.13: Images from left and right cameras showing identified points including one misidentified teat.

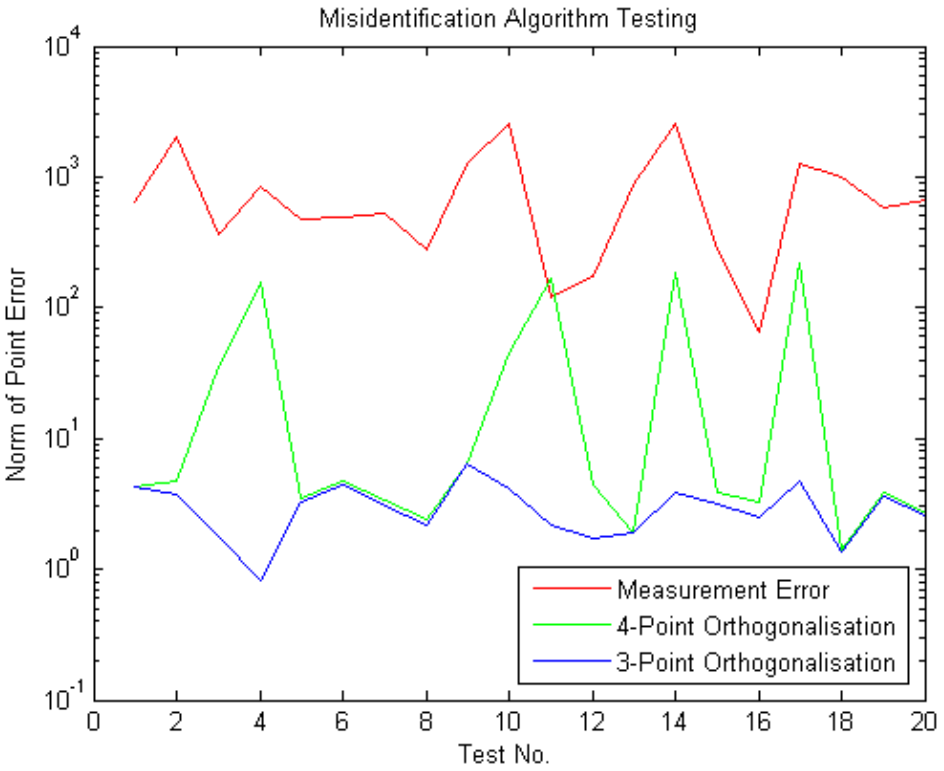


Figure 5.14: Misidentification algorithm error results.

Chapter 6

Discussion

6.1 Actuation System Component Accuracy

The results of the testing on the linear actuators are summarised in Table 5.1. The errors are very close to having a zero mean, with the largest being axis 4 with a mean offset of just half a millimetre. The standard deviations of the errors are also very low with the largest being less than a quarter of a millimetre. This indicates that the contribution of the end-effector linear actuators to the overall error of the actuation system is really quite small.

Similar data is presented in Table 5.2 regarding the revolute axes. The axis with the largest error is axis 3. This has a mean error over 4 times larger than the next largest mean error (axis 2). Axis 3 also has a standard deviation far in excess of the other axes.

The data implies a problem with axis 3. The four axes have an identical software setup and control system with many shared elements. It is believed that a problem with this would result in a problem on all 4 servomotors. Similarly, any problems with the measurement of the positions of the arm after each move or with the calculations used to find the move angle resulting from the command would be shared across the four axes. The problem then must lie with the hardware.

The backlash in the servo motors was also measured. They were found to have similar

amounts of backlash ranging from about 1.4° to 1.75°, as presented in Table 5.3.

Figure 5.1 show the results of the tests on the 6-axis robot arm along the x-axis and the y-axis respectively. The tests were performed within the workspace for this application. The values graphed are the error values, being the measured position of the point minus the theoretical position of the point. As the arm extends along the x-axis, that is, horizontally forwards away from the base of the robot, the accuracy deteriorates. This can be explained by the limitations of the position encoder resolution. The position encoders feed data back to the robot controller telling it the angle that particular joint is at. These encoders have a finite number of positions it can relay back. Thus, while the angular positional resolution is constant, the linear error will increase further along the link. In the case of the first joint of the robot, the 'waist' joint at the base, a small rotation will produce a larger linear translation of the end-effector, when the arm is fully extended. This is the case in the current tests. When the arm is fully extended, a small error in rotation of the first joint produces an error on both the x- and y-axes. This effect applies to all joints and on a serial link manipulator, the errors will accumulate. It is worth noting the fact that the mean errors measured on the robot (Table 5.4) are significantly larger than the repeatability value of ±0.1mm quoted for this model of robot on the manufacturer's website [71]. Another possibility is that the errors are due to inaccuracies in the transformation matrix mapping Microscribe points to robot coordinates. The definition of the reference frame used by the Microscribe is described in Section 2.1.5. If the assumptions made in setting up this transformation are wrong or if the Microscribe itself were to shift relative to the robot then systematic errors similar to those shown in Figure 5.1 would be seen.

The information in Table 5.4 represents the mean and standard deviations on each axis for the two tests (illustrated in Figure 5.1) combined. These show the accuracy of the robot over the workspace used by the system to be an order of magnitude smaller than the overall goal of the system.

6.2 Stereo Vision Pixel Level Accuracy

The errors generated by imposing a one pixel correspondence mismatch are presented in Figure 5.4. These values are presented in the coordinate frame of the vision system. It would be tempting to describe these errors in terms of the equations describing the basic concept of triangulation, however this system is a little more complex. As detailed in Section 3.2.2 the Bouguet stereo vision toolbox used to calibrate the camera pair includes a model for lens distortion making the overall model more complex than simple pinhole cameras. This must be kept in mind when relating the information presented in Figure 5.4 to the basic triangulation problem. There is certainly an error introduced when an adjacent pixel is selected, however these appear to be quite small. As expected, shifting the x-direction pixel produces little or no effect in the y-direction. Likewise shifting the y-direction pixel has minimal effect in the x-direction. The other expected aspect is the constant error on the z-axis when the x-axis pixel is changed. One would expect the depth error to increase when the corresponding points on the two images are separated. What is unexpected is the inconsistency of the changes on the z-axis when the y-axis pixels are changed. The change in pixels values in the y-direction are not expected to have the same influence as changes in the x-direction. This is because the separation of the cameras is far larger in the x-direction than in the y-direction. The same pattern (constant positive error) as the case for mismatch on the x-axis would be expected but it seems that for the first three points taken the triangulation actually estimated the points to get closer to the camera when one of the pixel values was increased. The z-axis errors caused by pixel-level mismatch in the y-direction are however far smaller than those created by x-direction mismatch. The changes in the value on the x-axis when the x-direction pixel is changed seem to show quite a systematic pattern to the error. The error has a linear pattern moving from negative to positive as the selected points move along the cameras x-axis. Looking more closely at the 3D coordinates of the points used, the error goes from negative to positive around where the points cross the zero on the x-axis. This is

a characteristic of the errors associated with the vision system that will be encountered again in Section 6.3.2.

6.3 Actuation and Sensing

In this round of testing, information was gathered on the accuracy of the robot and end-effector in positioning the milking cups below the teats and on the accuracy of the measurements taken by the vision system. The accuracy of the overall actuation system is discussed first.

It must be mentioned that, at the start of the project, a maximum error of about 5mm was decided on for the system. This came from anecdotal evidence from dairy farmers that the opening of the milking cup need only be at approximately this distance from the end of the teat for the vacuum to pull the teat in and successfully apply the cup.

6.3.1 Actuation

The positioning errors of the system are defined as the positions achieved by the robot minus the teat positions as measured by the Microscribe. These measurements are given in robot coordinates. The errors for the x-axis on teat 1 are plotted in Figure 5.5. There does not appear to any systematic pattern to these errors. Similar plots for the other axes on other teats also showed random errors.

The mean and standard deviations of the errors on each axis of the coordinate frame, presented in Table 5.5, show that the actuation system has a limit to its accuracy. While this is to be expected, the information presented quantifies this limit. The data also shows that the mean is non-zero and negatively biased. This may be due to backlash in the servomotors or a combination of the sources of errors discussed in later in this section. This information can be compared to the expected error in the system. This expected error is found by combining the errors in each direction contributed by the constituent parts of the system i.e. the values given in Table 5.1, Table 5.2 and Table 5.4. The

cumulative errors have been calculated as follows. The displacement in the x-direction is the sum of displacements of the robot, x_N, the linear actuators, x_L, and a component due to the rotation of the revolute axis, θ. This is expressed in Equation (6.1).

$$x = x_N + x_L + h * \cos\theta \tag{6.1}$$

The overall error, w_x, is expressed as follows.

$$w_x = \sqrt{\left|\frac{\delta x}{\delta x_N}\right|^2 w_N^2 + \left|\frac{\delta x}{\delta x_L}\right|^2 w_L^2 + \left|\frac{\delta x}{\delta x_\theta}\right|^2 w_R^2} \tag{6.2}$$

$$= \sqrt{w_N^2 + w_L^2 + h^2 * \sin^2\theta * w_R^2} \tag{6.3}$$

The value of w can be any measure of the error, in this case, twice the standard deviation, $2 * \sigma$, is used. The overall error in the y-direction is found similarly and is given in Equation (6.4). Here, the linear axes move only in the x-direction, their contribution to the error in the y-direction is 0. Also the contribution of the rotation is modified since it is in a perpendicular direction.

$$w_y = \sqrt{w_N^2 + h^2 * \cos^2\theta * w_R^2} \tag{6.4}$$

Table 6.1 shows these expected cumulative errors in the actuation system. The revolute errors are based on an assumption of the revolute axes being at a 90° position when the x-axis cumulative errors are calculated and at 0° position when the y-axis errors are calculated. The largest tangential errors due to the servo motors are calculated as $e = h * \cos\theta$ where h is the length of the respective revolute arm length and θ is the mean revolute error associated with that axis (taken from Table 5.2).

The total 2σ error (i.e. twice the standard deviation, ∼95% of measurements should fall within $\pm 2\sigma$ of the mean) is larger in the x- and y-directions for teats 1 to 3. This means that the total measured errors actually show less variance than expected on these

Table 6.1: Propagation of errors through the actuation system.

Teats	1		2	
Axis	x	y	x	y
Robot error, 2σ (mm)	0.5	0.37	0.5	0.372
Linear error, 2σ (mm)	0.25	0	0.14	0
Revolute error, 2σ (rad)	0.037		0.04	
Total error, 2σ (mm)	2.47	2.43	3.44	3.42

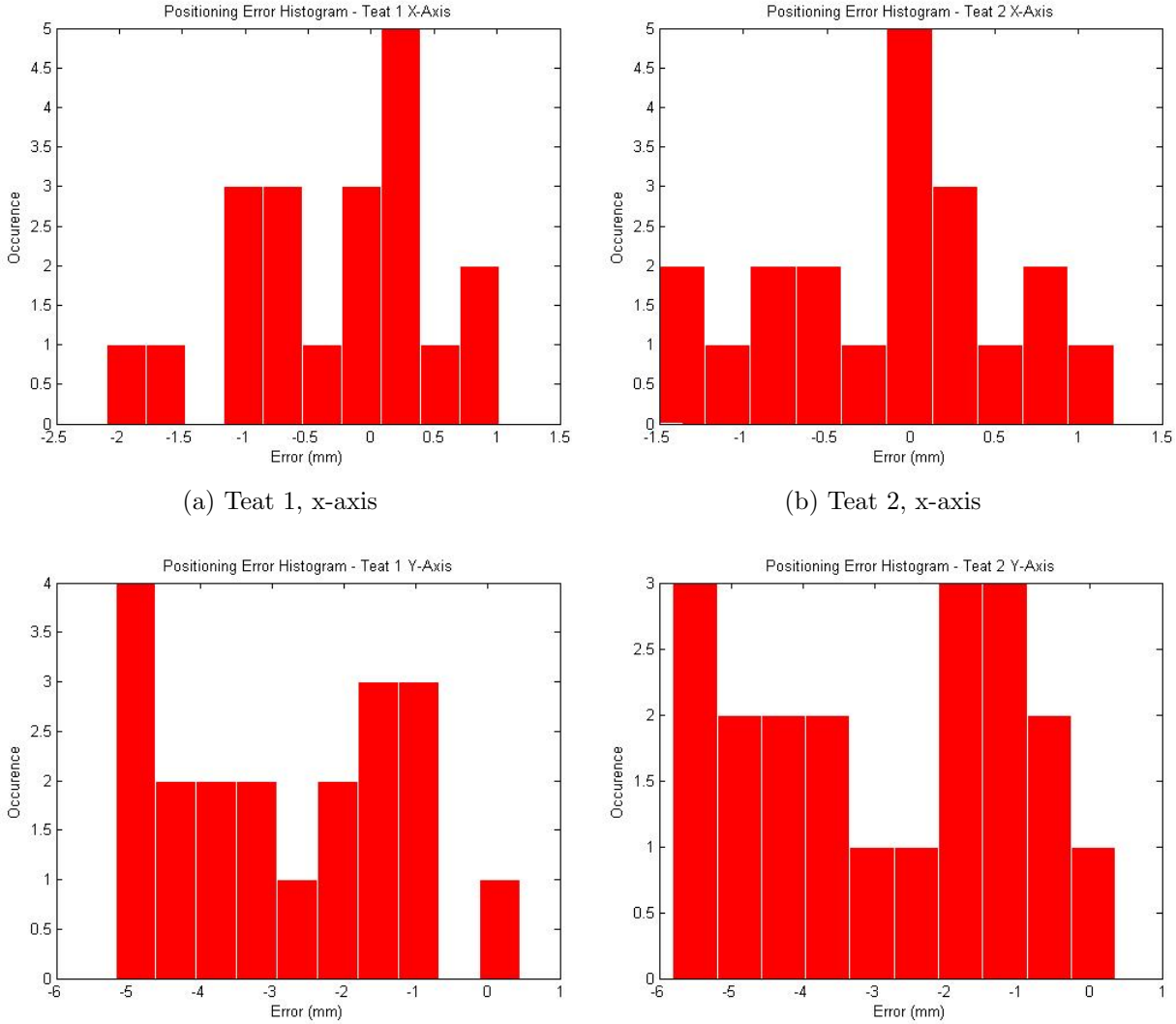
Teats	3		4	
Axis	x	y	x	y
Robot error, 2σ (mm)	0.5	0.37	0.5	0.372
Linear error, 2σ (mm)	0.23	0	0.48	0
Revolute error, 2σ (rad)	0.09		0.006	
Total error, 2σ (mm)	5.88	5.86	0.86	0.631

three axes. The fourth teat shows more variance due to measured mean error from the linear actuator being much larger for this axis. From Table 5.5, the x- and y-axis error values for teat 4 seem similar to the other teats. In Table 6.1, the estimated errors are significantly lower than the others. This is due to the small standard deviation seen on the revolute axis errors. Teat 3 also has particularly large estimates. This is down to the large errors that were seen on the revolute actuator associated with that teat. Sources of the errors seen on the actuation system are discussed in Sections 6.3.1.1 - 6.3.1.4.

Only the errors in the x- and y-axes (horizontal plane) are presented in Table 5.5 and Table 6.1 since the z-axis positioning accuracy is a lot less critical. As explained previously, the teats are assumed to have the same position on the vertical axis and the cups are placed directly below the teats and moved upwards into position. We know that in practice the teats have different heights and so larger errors on the z-axis of the robot coordinate frame are tolerated.

Figure 6.1 shows histograms for the x- and y-axes of teats 1 and 2. Since each graph represents only twenty measurements it would not really be expected to show a smooth bell curve. Here, the data has been divided into ten even groups i.e. ten vertical bars.

While the x-axes do appear to be approaching normal distributions, this is less evident on the y-axes. These distributions show a tendency for errors at the outer bounds to occur.



(a) Teat 1, x-axis

(b) Teat 2, x-axis

(c) Teat 1, y-axis

(d) Teat 2, y-axis

Figure 6.1: Histograms of measured positioning errors.

There are several possible causes for positioning errors in the system. Discussed below are the following issues:

- errors due to the Microscribe
- gear backlash

- revolute arm alignment
- kinematic constants

6.3.1.1 Microscribe

One element of the system that is used in all aspects of the system, including the vision system is the Microscribe, used as a ground truth. This piece of equipment has a positional resolution of 0.13mm and positional accuracy of 0.38mm. Both of these values are mean values and are quoted in the Microscribe user guide. A series of ten measurements of the same point (several centimetres from the base of the Microscribe) were taken to verify these values. The probe was replaced in its home position after each measurement. The mean and standard deviations on each axis are presented in Table 6.2. It was believed that due to encoder resolution that errors may be greater at full extension, as with the robot. The process was repeated using a point near the full extent of the probe, 500mm from the first point. The mean and standard deviations of these measurements are shown in Table 6.3. These two sets of results show that the mean of the error tends to be larger the greater the distance from the base.

Table 6.2: Mean and standard deviations of Microscribe measurements of a point close to its base.

Axis	x	y	z
Mean, (mm)	-0.01	0.11	0.08
Standard Deviation, σ (mm)	0.11	0.1	0.1

Table 6.3: Mean and standard deviations of Microscribe measurements of a point 500mm further from the base than that in Table 6.2.

Axis	x	y	z
Mean, (mm)	-1.2	0.04	-1.25
Standard Deviation, σ (mm)	0.09	0.04	0.11

6.3.1.2 Backlash

Of the three main elements of the actuation system (robot, stepper motors and servo motors) only the servo motors show appreciable backlash. This is due to the use of a planetary gearbox in which the backlash between gears accumulates as force propagates through the system of gears. The errors measured on the revolute axes do not allow for backlash which was measured as being about 1.6° (Table 5.3). This translates to a tangential error of up to 1.8mm on teat axes 1 and 3 (65mm revolute arm) and up to 2.4mm on teat axes 2 and 4 (85mm revolute arm). The measurements of the positions attained by the overall system include backlash errors.

6.3.1.3 Revolute Arm Alignment

An issue with the home positions of the revolute axes is believed to have caused a constant offset rotation of the revolute arms. The revolute axes each have a home position sensor. This is fed back to the servomotor controller and allows the motor to be commanded to a home position. The revolute arms are clamped onto the shaft turned by the motor. However there is no mechanical way of ensuring that the arms are lined up exactly to the home position. The arms do not need to be in line with the home position sensor, only to be parallel to the linear axes of the end-effector. This was done by visually aligning each arm to the chassis of the end-effector. A steel square was used to try to line up the revolute arm to be parallel to the end-effector chassis. However it was not possible to find a position where this could be guaranteed due to the geometry involved.

6.3.1.4 Kinematic Constants

In the process of improving the accuracy of the actuation system, the kinematic constants of the end-effector were measured and compared to the design values (taken from the 3D model used to design the end-effector). It was found that there was a difference of up to several millimetres between the design values and the values measured using the Microscribe. These differences are believed to result from the assembly of the end-effector.

Most parts of the end-effector are held in place by multiple bolts where the bolt holes are larger than the bolts themselves. There is thus a certain amount of play between parts and when bolts are tightened, the parts may not be aligned exactly as designed. For example, each linear arm is held in place by two pairs of grooved roller bearings that grip a track bolted to the side of the arm. If the lower or upper two of these rollers are tightened first, the arm can be held slightly higher or lower in the chassis respectively. Similarly, tightening diagonal rollers first can cause the arm to be slanted. The values measured using the Microscribe were used in the inverse kinematics software.

6.3.2 Sensing

The results of the testing of the accuracy of the vision system are summarised in Table 5.6. These values represent errors on the axes in the stereo vision system's own coordinate frame. It can be seen that the mean errors on each of the x- and y-axes come well within the 5mm limit. These results show that the cameras are more than capable of finding the positions of the teats with enough accuracy to place the cups.

It must be noted that the errors presented have non-zero means. This would indicate some systematic problem causing a bias in the measurements by the vision system or by the Microscribe. There are several possible explanations for this.

One is the fact that when the tip of the Microscribe is inserted into the hole on the endpoint of the teat, the very tip is several millimetres inside the hole. The tip can also move around inside the hole depending on the angle at which it is held. When measuring the positions of the teats it was also tried to keep the Microscribe tip vertical and centred however this is purely down to the user and an error of up to 1mm in any direction may be encountered given the diameter of the hole is 2mm.

The non-zero mean can be seen as a constant offset. A major contributing factor to this could be an error in one or more of the transformation matrices. The techniques used in finding these transformations were described in Sections 3.2.4 and 2.1.5. Any error in the first three columns of a transformation matrix (which describe the rotations

between coordinate frames) will produce an error that increases with distance from the origin. An error in the fourth column (which describes the position of the origin of one coordinate frame relative to the other) however will be seen as a constant offset. This could provide an explanation for the non-zero means of the errors.

Another factor which could produce a biased error is user error. As illustrated in Figure 5.2 the lighting conditions can change the appearance of the hole in the endpoint of the teat. The centre of the hole can be more difficult to identify and if there is a glint of light on the edge of the hole, the user may perceive the centre of the hole as being in a position with a constant offset from its actual position. Similarly, if the teat is at more of an angle, the hole may appear as an ellipse.

The measurement errors are also presented in Figure 5.6 where the errors are plotted in the order in which the measurements were taken. The data has been separated to show the errors on each teat in the four graphs with the errors on each axis presented on separate subplots.

The data in these graphs appears to have an almost periodic quality. The reason for this is that the measurement index that the errors are graphed against represents the order in which the measurements were taken chronologically and not sorted according to any criteria. Data points that appear to have similar error values are multiple measurements of the same point in space, keeping in mind that for each configuration of the teat cluster the teats were measured six times each. Thus there appears to be a correlation between measurements of the same points. In order to examine this correlation further, the data was sorted according to position along each axis. Figure 5.7 shows three graphs plotting the sorted error against the position along each axis.

These plots show that the errors are highly correlated to position. The errors on both axes go negative approximately as the position passes the principal axis. This suggests that the x- and y-axes of the vision system coordinate frame are not parallel to those of the Microscribe coordinate frame. It appears that transformation matrix between the coordinate frames of the stereo vision system and the Microscribe has an error that

effectively rotates the transformed coordinates about the z-axis. This is evidenced by the fact that the z-axis does not show a similar pattern and is thus unaffected. Figure 5.7c also shows that the standard devation of the error gets larger as the z-axis coordinate increases. This is consistent with the errors expected from a stereo vision system such as this and as verified in the tests described in Section 5.2.

It is hypothesised that errors caused by choosing a nearby pixel had become a part of the system when the robot to camera transformation matrix was found. In this procedure, described in more detail in Section 3.2.4, a point on the end-effector moved to a series of positions in the vision systems field of view and measured by the vision system was used to construct a transformation. If the measured points were not accurately selected, i.e., if a nearby pixel was selected instead of the optimal one, then the transformation itself would have errors incorporated into it. If these errors were in the first three columns (dealing with rotations of the axis of one frame relative to the other), then an error would be seen along that axis that would increase with distance from the origin.

As represented in Figure 5.4, the errors on the y-axis due to x-axis pixel mismatch are consistently very close to zero but on the z-axis there is an error of almost a millimetre. Given a typical transformation matrix ${}^{R}T_{C}$, this 1mm error can be converted as a point into robot coordinates and the origin subtracted to give the equivalent error in the robot coordinate frame.

$$ {}^{R}T_{C} = \begin{bmatrix} -0.063 & 0.766 & 0.644 & 1765.7 \\ -0.984 & 0.007 & -0.202 & 176.6 \\ -0.092 & -0.671 & 0.739 & 108.6 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6.5} $$

$$ error = \left({}^{R}T_{C} * \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right) - {}^{R}Trans_{C} \tag{6.6} $$

Where $Trans$ is the translation vector between coordinate frames i.e. the fourth column of the transformation matrix. This gives the error as [0.6442; -0.2019; 0.7391]. The error on the camera z-axis shows up mostly on the robot x- and z-axes because the camera z-axis points roughly between the x- and z-axes of the robot coordinate frame. This means that if the point selected is one pixel over on the camera x-axis, then an error of 0.6mm will be seen on the robot x-axis and 0.7mm on the robot z-axis. Analysis of the data collected shows that point selection has a standard deviation of about one pixel meaning around 95% of points selected will be within ± 2 pixels of the best points.

If the points selected in setting up the robot to camera transformation are out by just one pixel, the points selected to define each axis could be offset in the camera z-direction by 1mm. These are points 100mm from the origin of a frame that was translated to the camera frame (see Section 3.2.4). If this error was on the point defining one axis of the camera frame, points on that frame would show an error increasing with distance along the axis.

The transformation matrix can be analysed more closely to check for errors. The first three columns describe the rotations of the destination coordinate frame relative to the source frame. More specifically, these columns describe unit vectors lying on the axes of the destination frame relative to the source frame. For an orthogonal frame each of these vectors must be equal to the cross product of the other two. This is easily verified.

$$error_x = \| {}^R T_{Cx} - ({}^R T_{Cy} \times {}^R T_{Cz}) \| \tag{6.7}$$

$$= 0.0968 \tag{6.8}$$

$$error_y = \| {}^R T_{Cy} - ({}^R T_{Cz} \times {}^R T_{Cx}) \| \tag{6.9}$$

$$= 0.0327 \tag{6.10}$$

$$error_z = \|{}^R T_{Cz} - ({}^R T_{Cx} \times {}^R T_{Cy})\| \tag{6.11}$$

$$= 0.0923 \tag{6.12}$$

The larger errors on the x- and z-axes compared to the y-axis corresponds with the fact that the x- and z-axes are affected more by a camera z-axis error. The level of these errors also suggests more than a one pixel correspondence mismatch during calibration.

It should be noted that the inaccuracies described here are not the only sources of error in relation to the vision system. Stripping the vision system of all extraneous elements and looking at the 3D coordinates produced by the triangulation routine from pixel values, and the errors that are produced when these pixel values are changed, it is believed that the source of the characteristic errors in the system have less to do with transformation matrices and more to do with the intrinsic and extrinsic parameters of the camera as found by the calibration procedure. Indeed, during the course of the research, the calibration procedure was carried out several times with wildly varying results, some of the errors resulting in large constant offsets (up to 7mm) on certain axes, and some with almost zero means and standard deviations of as little as 0.3mm. It was beyond the scope of this project to investigate different calibration routines as this is quite a large area in itself. A lot of time could be spent comparing alternate camera models and calibration methods to find the most accurate parameters. This system requires quite a high level of accuracy due to the fact that any error in measuring a point will be propagated through the system as part of the transformation matrices. There are anomalies in the camera measurements, but these would require further research to fully understand and deal with.

Despite the inaccuracies described, the errors displayed by the vision system in these tests are still below maximum error goal of 5mm for the system.

6.4 Moving Vision System

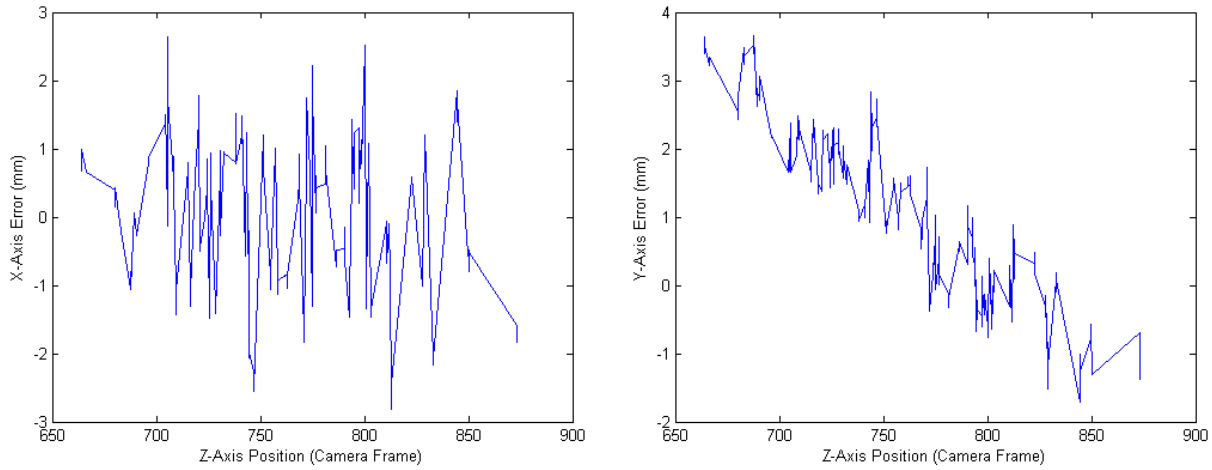
In the next set of results the cameras were fixed to the end-effector which was moved incrementally towards the teats and measurements were taken at each step. The absolute values of the mean of the errors are shown in Figure 5.8. The data for the four teats has been separated into the four graphs. Each graph shows the absolute mean error for the twenty repetitions of the test, plotted against the step number for each axis of the camera coordinate frame. This allows us to analyse the mean error on each axis at each step. By seeing each axis separately we should get an insight into the characteristic errors of the measurements produced by the vision system.

The vision system measurement errors were expected to continuously decrease as the cameras approach until they reach the distance of optimal focus. Beyond this point the errors are expected to increase as the cameras get closer and the sharpness of the images deteriorates. However, the results, shown in Figure 5.8, were not as expected.

The most striking thing about these graphs is the fact that the z-axis errors dont appear to show any of the continuous improvement that was expected as the cameras approached the teat cluster. The depth value (z-axis) errors remain high. On the x-axis the expected pattern of errors is seen, that is, reducing to a point and then increasing as images go out of focus again. The point at which the x-axis values are at their best does not correlate with the position of the teat. The focus of the cameras was adjusted to allow all four teats to be in focus at the same time so that they could be measured simultaneously. The four teats should then be at the point of best focus at the same time. At the least 1 and 3, which are generally closer to the cameras, should have the same point of best focus. Similarly for teats 2 and 4, which are further away. However this does not happen. Teats 2 and 3 reach best focus around step 4, teat 1 reaches it around step 3 and teat 4 around step 5. Meanwhile, the y-axis starts with a larger error than either of the other two axes and shows a sharp reduction in error up to the last measurement, with no deterioration after the point where the x-axis errors start to get

worse.

Since these results plot the relationship between the errors on each axis and the approximate z-axis positions of the points measured (in the vision system coordinate frame), it is useful to return to the data presented in Figure 5.7. The data from these earlier tests can be used to show any correlation between the errors on each axis and the z-axis positions, similar to Figure 5.8. This re-organised data is shown in Figure 6.2. This shows the z-axis position graphed against x- and y-axis errors only, since the same graph is presented for the z-axis in Figure 5.7c. Figure 6.2a does not appear to show any systematic relationship and this bears out what is seen in Figure 5.8. Figure 6.2b however, shows that the error on the y-axis rapidly goes from positive to negative as the point moves further from the vision system. This would appear to corroborate the pattern seen in Figure 5.8 where the y-axis error is rapidly changing. The difference is that Figure 5.8 covers a larger z-axis range.



(a) X-axis (b) Y-axis

Figure 6.2: Vision system measurement data from Figure 5.7 plotted against z-axis position.

Another issue that was noticed during the testing was that after the cameras reach and pass through the point of best focus and continue to approach the teat cluster, the teats continue to become larger in the images. This provided a larger target to select the endpoint of the teat. Although the hole at the endpoint was out of focus it was larger so

it was believed that the centre of the hole could be selected with more accuracy leading to the errors improving.

It can also be noted that, as the results of the pixel level accuracy testing showed, if the endpoints are not selected accurately, the largest errors will be seen on the z-axis. As the images of the teats come into focus, the endpoints should be selected more accurately and the z-axis errors reduce. This does not happen.

One explanation for the behaviour seen is that as the cameras approach the teats, the endpoints appear to separate and move closer to the edges of the image. Here, they are more prone to errors arising from lens distortion. The only other possible explanation for the behaviour in this test is in the calibration. Because the cameras were moved to a new position (i.e. on the end-effector) it was necessary to re-calibrate them. It is believed that this calibration produced different characteristic errors to those seen previously and subsequently when the cameras were re-calibrated for use on the floor.

6.5 Kalman Filtering

6.5.1 6 Steps

The test was originally performed with twenty runs of only 6 measurements. The results of one such run is presented in Figure 5.10. These values are errors in the robot coordinate frame. While the filter does appear to smooth the results it does not consistently reduce the errors on all teats. The results for the y-axis, where most movement occurred in the test, seem to show the Kalman filter improving the results substantially but then getting worse. Ideally we would see the filter converge and maintain the improved results but there appears to be a settling phase that does not conclude within the time frame of the run. Six iterations are not enough to analyse the behaviour of the system. It was also known from the simulations that the Kalman filter could have a settling time in excess of 6 time intervals depending on the initial parameters used. The decision was taken based on this to repeat the test with more data points.

6.5.2 21 Steps

It was decided to perform a test that would run over the course of 20 movements of the teats. Including the initial position, this would provide 21 data points. Another modification was to use steady-state Kalman gains. This is essentially where the values to which the Kalman gains converge in one test are used as the constant Kalman gains of another. The idea is to avoid any settling time in the error by using an initialisation phase. This is discussed more in Section 4.4.

Figure 5.11 shows the results from the both the first run with time-variant gains and the second run with steady-state gains (using the same set of measurements). These are the errors of values given in the robot coordinate frame.

While there are significant improvements on some axes (e.g. teat 2, x-axis), some are inconsistent (e.g. teat 1, x-axis). Others show hardly any change at all (e.g. teat 2, y-axis). These results were not deemed satisfactory. The main reason for this is that better results were seen in simulation. It was believed that similar results could be achieved on the experimental data. The improvements seen on teat 1, y-axis and teat 2, x-axis also indicate that significant improvements can be made. The use of steady-state Kalman gains is beneficial. However another possible approach would be to use more data points in a longer test using a time-variant filter. The result of this should show an initial settling time followed by significant improvements over the non-filtered measurement data.

6.5.3 101 Steps

The measurements recorded by the vision system and the Microscribe were input to a Matlab script and the Kalman filter applied to the data. Running the filter offline with experimental data allowed a lot more experimentation with parameter values such as the covariance matrices and the initial state estimate values. For the first set of results shown in Figure 5.12a, the initial state estimates, denoted $\hat{x}_{k-1}$, were set equal to the measurements of position, y_{k-1}, at the second step of the process and the velocity, $\hat{\dot{x}}_{k-1}$,

calculated from the first two measurements (as described in Section 4.4.2). Although this was not always very accurate (the low velocity meant that measurement noise could easily cause very inaccurate velocity estimates, resulting in the overshoot in error magnitude seen in the first 10-15 iterations in Figure 5.12a), it improved the settling time caused by setting the initial estimates to zero.

The measurement covariance matrix, Q, was found by passing the measurement errors found using the cameras and the Microscribe to the cov() function in Matlab. This produces a matrix of very small values ($\sim 10^{-2}$ - $\sim 10^{-3}$) with larger ones ($\sim 10^{-1}$) in key places. These larger values are found where autocorrelation is implied for x- and z-values but interestingly not for y-values. Larger values are also found implying cross-correlation between the x- and z-values of each teat.

The process noise covariance matrix is defined as follows. The state transition equation of the system is given as $x_k = Ax_{k-1} + Bu + w_k$, where w is the zero-mean uncorrelated process noise [29]. The covariance of the process noise at time k is then $R_k = E[w_k w_k^T]$ (from the definition of covariance) where $w = B * \sigma_R$ and σ_R is the standard deviation of the process noise. This basically says that the process noise is an acceleration that acts on the system through the matrix B. The B matrix usually describes the effects of a command input, u, however in this system there is no command and u = 0. In practice the process noise covariance can be difficult to find and is often tuned to a value that gives good results [56]. In this case a value of 0.0001 was used for the process noise standard deviation. The state estimate covariance, P, was initially set equal to the identity matrix. Through experimentation with values it was found that although the identity matrix provided no noticeable settling time the results of the Kalman filter were not as good as expected. This matrix was then multiplied by a scalar value of 0.001 which gave significant improvement in performance. The drawback was that the Kalman filter had quite a long settling time. This value can be tuned to give the desired performance. If settling time is not an issue and only the last measurements are to be used to estimate the teat positions, the estimate accuracy can be improved by reducing this value. What

this is essentially doing is telling the Kalman filter not to put too much faith in the initial state estimates. As the Kalman filter runs and new, more accurate estimates are generated, the P matrix is recalculated and the filter trusts its own estimates more.

Table 5.8 shows the standard deviations of the last 40 measurements of each axis of each teat, before and after filtering. Kalman Error (i) refers to this current setup of the Kalman filter. The biggest improvements are on the x- and z-axes but the y-axes also show improvements. The graphs in Figure 5.12 show the errors of the measurements and the Kalman estimates for the y-axis of teat 1. The three graphs represent the results from the three configurations of Kalman filter summarised in Table 5.7.

There are two other possible ways to formulate the process noise covariance matrix, R, both using measurements taken from the system. The process noise covariance is a measure of the error in the system model used to predict the new state of the system. The Kalman filter uses this and the measurement data to provide an optimal estimate of the true state. Since the process cannot be observed directly and can only be seen through measurements, the covariance is generally quite difficult to find. In this system, definitive ground truth measurements are taken with the Microscribe. Although this is merely another form of measurement and will have its own measurement noise, it is believed to be more accurate than the vision system. The assumption has been made throughout the project that these values are the true values of the positions of points measured. The process noise can then be measured based on this assumption.

The first way this is done is by repeatedly measuring a set of points (the endpoints of the teats) with the Microscribe. The teats are measured in the same position each time, but between each measurement, they are moved away and back into position via the slider mechanism. The purpose of this is to introduce the noise on the position associated with the process of moving the teats. The idea is that the process noise is essentially being replicated only since the position is meant to be the same each time, i.e. the movement is zero and all that is left is the noise. This gives a series of measurements of the same positions that will have a standard deviation corresponding to the process noise. The

covariance of this set of measurements will provide information about the noise on the positions but not the velocities. The velocities are found by calculating the differences between successive measurements. These velocities are concatenated with the positions and the covariance of the resulting matrix is calculated. This results in the 24 element square matrix required.

This formulation of the process noise covariance is used in another filter with the same measurement noise covariance matrix used previously. The results are graphed in Figure 5.12b and the standard deviations of the resulting errors are shown in Table 5.8, referred to as Kalman Error (ii).

This change in the formulation of the process noise results in a reduced settling time on all axes and a simultaneous increase in the standard deviations of the errors. The covariance matrix produced in this way generally has very small values showing little or no relation between the change in values of the variables due to the process. This is perhaps not surprising since the teats were stationary when the values, from which the covariance was calculated, were taken. In other words, since none of the states were changing, it is unsurprising that there appears to be no correlation between the changes of the states. What was basically measured here is the measurement noise of the Microscribe and the process noise for a situation where the process is practically inactive.

Another way to formulate the covariance of the process noise is to think of it as the difference between the state estimated by the system model and the actual positions recorded (again by the Microscribe). This should quantify the accuracy of the system model as compared to data from the Microscribe. In the development of the Kalman filter simulation, the simulated measurements generated by the system model were compared to the actual measurements and were found to be very close (Section 4.2.2). It should thus be possible to compare state estimates generated by the system model and the true values of the states as measured using the Microscribe. The covariance of this set of error data is the process noise error covariance.

To implement this, a set of data was generated by the system model using the initial

positions from the experimental data as a starting point. This data is the same set of values that was compared to the Microscribe measurements in Section 4.2.2, and does not include a term modelling the process noise. This is so that the prediction is noise free and the error values include process noise [72]. The conclusion from that stage in the development of the simulation was that the simulated data was a very close approximation to the actual values. The difference between the simulated data and the data measured by the Microscribe was found and the covariance of the result calculated. This covariance matrix was found to have an upper left quadrant (values dealing with the correlations between positions) similar to the measurement covariance used in the set of results referred to as Kalman Error (ii). This was where Matlabs cov() function was applied to the difference between the vision system and Microscribe measurements. The similarity is that larger values were found to correlate the x- and z-axes of each teat while the y-values which were a lot smaller implied they did not auto correlate. The reason for the similarity seems to be that because some variables are not changing, they appear to be correlated i.e. have related behaviour. The approach does not provide the best results. If positions were used where the teats were moving in all directions independently of each other, then a better covariance matrix would be found. The results for the Kalman filter using this final formulation of the process noise covariance are shown in Figure 5.12c and the standard deviations of the errors are denoted Kalman Errors (iii) in Table 5.8.

These results are noticeably worse than previous versions of the Kalman filter. The data in Table 5.8 shows that although it is worse it is still an improvement over the raw measurements. Reviewing the data in Table 5.8, the filter that produces the lowest estimate standard deviations on each axis is the Kalman (i). As Table 5.7 reminds us, this uses a process noise covariance matrix, R, constructed using an arbitrary standard deviation $\sigma = 0.001$.

The parameters used in this version of the Kalman filter result in a longer settling time but also the smallest standard deviation over the final measurements. It must be noted also that the vision system errors and hence the Kalman state estimates have non-

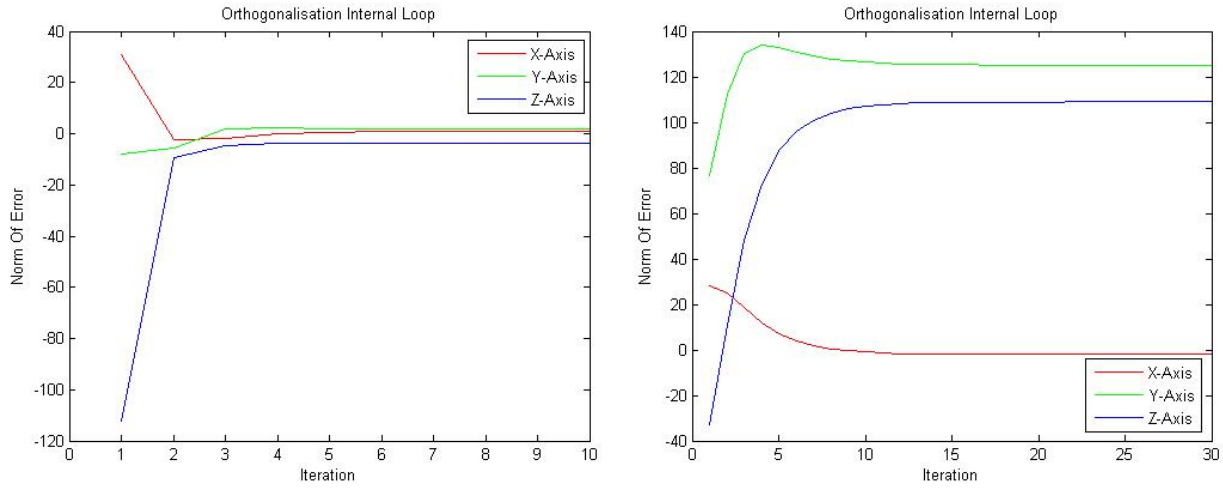
zero means. One of the key assumptions of the Kalman filter is that the process noise and the measurement noise are both Gaussian random vectors with zero mean, however it is possible to add in an extra constant additive term to the state transition equation and the measurement equation describing the system [58]. This will be a constant offset to shift all of the values to have zero mean. Caution would be advised as it is possible the mean of the noises is not a constant and there could be drift as a function of time of any of the state variables or of any combination of these.

6.6 Handling of Misidentified Teats

Two algorithms were developed to send the correct target positions for all four teats to the robot when one of the teats has not been correctly detected. The graph in Figure 5.14 shows the results of the tests on the two algorithms. The horizontal axis represents the series of twenty tests in the order they were performed. The vertical axis shows the Euclidean norm of the error in millimetres. This uses a logarithmic scale since the original errors and some of the results are orders of magnitude larger than the smallest errors. The graph shows the norm of the error of the misidentified teat. This is the position calculated by selecting the wrong point in the image. The graph also shows the norm of the error of the point estimated by the four-point routine (Section 4.6.3.1) and the estimated point resulting from the three-point algorithm (Section 4.6.3.2).

In all but one case (test 11), the four-point algorithm drastically improves the position of the misidentified teat. However, this improvement is not enough to be useable in several cases (e.g. tests 4, 11, 14 and 17) as the resulting positions are still out by up to 220mm. It is believed that in the iterative process of the four-point routine, the algorithm converges on an ortho-normal coordinate frame that is not the optimal solution. Figure 6.3a shows the four-point algorithm iteratively improving the coordinates of the misidentified teats, in the first test. While this loop is run 100 times, the transformation matrix, and hence the coordinates, converge after about 6 or 7 iterations (note only the first 10 iterations

are plotted). Comparing this to Figure 5.14 we see that this produced a good estimate of the actual position of the teat. Figure 6.3b on the other hand shows Test 11 taking a lot longer to converge. As Figure 5.14 shows, this test resulted in a coordinate that was out by 166mm. In Figure 6.3b, the y-axis component shows an overshoot, peaking around the fourth iteration. This would hint at the idea of the solution to this algorithm being non-unique.



(a) Test 1

(b) Test 11

Figure 6.3: Coordinates of point with largest error converging in the orthogonalisation routine.

Closer inspection of Figure 5.14 and the data that produced it shows that in only 2 of the 20 tests did the four-point algorithm give better results than the mapping algorithm. These are Tests 1 and 9 and the differences in the resulting errors are 0.0391mm and 0.0029mm respectively. Evidently there is not enough of an improvement to justify using the four-point algorithm over the three-point algorithm.

Chapter 7

Conclusions

The ultimate goal of this project was to develop a system that can detect the teats of a cow, measure their coordinates and retrieve and apply a set of milking cups to the teats, accessing them from between the hind legs. The current work has advanced upon this objective significantly. At the onset, the system had no integrated sensing system (although work had been done in this area). The hardware of the actuation system was in place and a low level control had been achieved over the actuation subsystems (robot, steppers and servos). The system was only capable of moving a single cup but was not capable of retrieving that cup. Teat positions were measured by manually moving the robot into position beforehand. The level of automation in the system was therefore minimal.

7.1 Summary of Work

This work has shown that it is possible to retrieve and apply all four milking cups simultaneously. This is an approach not seen on existing systems and is considered an important feature of a system that will be able to perform at a rate comparable to that of a human worker. This goal has been achieved through an analysis of the available working volume in the parlour (discussed in Section 2.1.1) and the design of an acuta-

tion system compact enough, yet flexible enough to reach most typical teat positions, as described in Section 2.1.3. The system has been designed for a carousel parlour, which is the type of parlour with highest volume, further increasing the potential productivity of a robotic milking system. The system has also been designed to for use in existing parlours, unlike all commercially available systems, where substantial renovation of the dairy parlour is required.

The work has further shown that a stereo vision system is capable of providing the accuracy required in detecting the teats to allow the milking cups to be placed reliably, shown in Table 5.6. This differs from existing laser-based detection methods in that time-consuming scanning procedures are not necessary. These results have been verified by experimentally showing the system to be capable of placing the cups on the teats using positions measured by the vision system. In an online system, this method will allow measurement of four teats simultaneously, further contributing to a decrease in application time. This is in contrast to the sequential methods used on commercial systems.

An investigation was conducted into methods of improving the accuracy of results using Bayesian filtering. The Kalman filter was found to generally reduce measurement errors when applied to this unique system (Fig. 5.12). Particle filtering also showed improvements but the investigation showed it to be too unreliable for use in this application. The algorithm could be modified but it was concluded in Section 4.5.2 that the even if reliability issues were overcome, the Kalman still provided better reductions in measurement noise.

Two novel methods of handling misidentified teats were developed and tested. These methods were designed for this specific application and work with the very limited set of points available (i.e. four). This makes them more suitable than existing robust identification techniques that work on clouds of points. One of the methods developed here was found to significantly outperform the other in reliably allowing a cup to be placed on the teat even in the case where a spurious object in the field of view of the

vision system has been identified as the teat and measured as such, Fig. 5.14.

7.2 Future Work

The most obvious next major step in the development of the overall system is to investigate teat identification that does not rely on human interaction. The system must be able to view the working area and locate the teats automatically. This will require substantial study of data gathered from working dairy parlours and image analysis. It will also include developing a technique of more accurately calibrating the stereo pair of cameras as this is believed to be the cause of inconsistencies in the results of testing. Such a calibration technique will need to provide consistent results so that if a calibration is repeated, similar noise characteristics will result. It is expected that the resulting automated system will involve some aspects from artificial intelligence, specifically pattern recognition. Given the popularity of the use of facial recognition algorithms in everything from security systems to social networking sites, the task of identifying the teats would seem to be within the means of current technology.

Substantial ground has been gained in this project. Cup application has been achieved and measured in a lab situation and tools developed to deal with measurement system errors. This has been done within the limits of the goal: developing a robot capable of applying teats between the rear legs of the cow. This is a goal which has yet not been reached by any automated milking system currently on the market.

The advances described here lay the groundwork for a novel robotic system capable of applying the milking cups to the teats of a cow in a way that outperforms existing, commercially available automated milking systems. The stated goals of the project outlined in Chapter 1 have been achieved and significant steps taken in further improving the performance of the system. This represents a step forward in the technology used in agriculture and specifically automated milking, paving the way for more economically viable farming.

Appendix A

Orthogonalisation Proof

Given an invertible n×n matrix A, find the orthogonal matrix R (with $det(R) = 1$) which minimizes $\|A - R\|^2$. Note: Here we have the standard inner product on $\mathbb{R}^{n^2}$ = all n×n matrices, given by:

$$\langle A, B \rangle = Tr(AB^T) \tag{A.1}$$

So:

$$\|A\|^2 = Tr(AA^T) \tag{A.2}$$

Let $SO(n)$ = the space of all special orthogonal matrices. To calculate the tangent space to $SO(N$ at the identity matrix, I, let $\gamma(t)$ be a curve in $SO(n)$ = with $\gamma(0) = I$. Thus ∀ t:

$$\gamma(t)\gamma(t)^T = I \tag{A.3}$$

To find the tangent to this:

$$\dot{\gamma}(t)\gamma(t)^T + \gamma(t)\dot{\gamma}(t)^T = 0 \tag{A.4}$$

At t= 0:

$$\dot{\gamma}(0)\gamma(0)^T + \gamma(0)\dot{\gamma}(0)^T = 0 \tag{A.5}$$

Since $\gamma(0) = I$

$$\dot{\gamma}(0) + -\dot{\gamma}(0)^T \tag{A.6}$$

That is, $\dot{\gamma}(0)$ is an n×n skew-symmetric matrix. Therefore, the tangent space to $SO(n) =$ at I is skew-symmetric.

Note: If $A =$ n×n symmetric and $B =$ n×n skew-symmetric, then:

$$\langle A, B\rangle = Tr(AB^T) \tag{A.7}$$

$$= Tr([AB^T]^T) \tag{A.8}$$

$$= Tr(BA^T) \tag{A.9}$$

$$= Tr(BA) \text{ (since A = symmetric)} \tag{A.10}$$

$$= Tr(AB) \tag{A.11}$$

$$= -Tr(AB^T) \text{ (since } B^T = \text{skew-symmetric)} \tag{A.12}$$

$$= -\langle A, B\rangle \tag{A.13}$$

$$\implies 2\langle A, B\rangle = 0 \tag{A.14}$$

$$\implies A \perp B \tag{A.15}$$

Thus (symmetric) $\perp$ (skew-symmetric). This implies that the space, normal to $SO(n) =$ at I is a symmetric matrix.

Our non-orthogonal n×n matrix, A is thus best approximated by $R \in SO(n)$ (i.e. $\|A - R\|^2 =$ minimum):

$$\implies A - B \perp T_R SO(n) \tag{A.16}$$

where T_R is the tangent space to $SO(n) =$ at R. Now our inner product $\langle A, B \rangle$ is invariant under the action of $SO(n)$ on the left (and on the right).

Indeed $\forall$ P $\in SO(n)$:

$$\langle PA, PB \rangle = Tr(PA(PB)^T) \tag{A.17}$$

$$= Tr(PAB^T P^T) \tag{A.18}$$

$$= Tr(AB^T P^T P) \tag{A.19}$$

$$= Tr(AB^T) \tag{A.20}$$

$$= \langle A, B \rangle \tag{A.21}$$

So since $\langle\ ,\ \rangle$ is invariant under the action of $SO(n)$, $(A - R) \perp SO(n)$ at R.

$$\implies (R^T A - I) \perp SO(n) \text{ at } I \tag{A.22}$$

Therefore, $R^T A - I \in$ symmetric matrices. Let:

$$R^T A = S \in \text{ symmetric matrices} \tag{A.23}$$

$$S^T S = (R^T A)^T R^T A \tag{A.24}$$

$$\implies S^2 = A^T R R^T A \tag{A.25}$$

$$\implies = A^T A \tag{A.26}$$

This is symmetric and so is orthogonally diagonalisable. Also, all eigenvalues ≥ 0.

Let $u_1, ..., u_n$ be an orthonormal basis for $\mathbb{R}^n$ consisting of eigenvectors of $A^T A$ with corresponding eigenvalues (which we can write as $\lambda_1^2, ..., \lambda_n^2$).

That is:

$$(A^T A)u_j = \lambda_j^2 u_j \tag{A.27}$$

If we put $P = [u_1, ..., u_n] \in SO(n)$

$$\implies P^T(A^T A)P = \begin{bmatrix} \lambda_1^2 & 0 & 0 & 0 \\ 0 & \lambda_2^2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_n^2 \end{bmatrix} \tag{A.28}$$

Now, $S^2 = A^T A$ and S commutes with S^2 (i.e. $SS^2 = S^2 S$), therefore S commutes with $A^T A$. Thus S and $A^T A$ are simultaneously diagonalisable. This means that $P^T S P$ is diagonal. But:

$$Sv = \lambda v \tag{A.29}$$

$$S^2 v = S(\lambda v) = \lambda S v = \lambda^2 v \tag{A.30}$$

Since $S^2 = A^T A$, then the eigenvalues of S are equal to the square root of the eigenvalues of $A^T A$. Therefore:

$$S = P \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \lambda_n \end{bmatrix} P^T \tag{A.31}$$

where $\lambda_j^2 =$ the eigenvalues of $A^T A$. So, for each j, there is a choice of sign $\pm\lambda_j$.

Let's now calculate the square of the distance between A and R.

$$min\|A - R\|^2 = min\|R^T A - I\|^2 \tag{A.32}$$

$$= min\|R^T A - I\|^2 \tag{A.33}$$

From A.23,

$$= min\|S - I\|^2 \tag{A.34}$$

$$= min\langle s - I, s - I\rangle \tag{A.35}$$

$$= min\ Tr([S - I][S - I]^T) \tag{A.36}$$

$$= min\ Tr([S - I][S - I]) \tag{A.37}$$

$$= min\ Tr(S^2 - 2S + I) \tag{A.38}$$

$$= min\ Tr(AA^T - 2S + I) \tag{A.39}$$

$$= min\ (\sum_i \lambda_i^2 - 2\lambda_i + 1) \tag{A.40}$$

$$= min\ \sum_i (\lambda_i - 1)^2 \tag{A.41}$$

So the minimum occurs when all of the eigenvalues, λ_i, are positive.

The orthogonalisation algorithm applies this in the form of Equation A.23, rearranged to give:

$$R = AS^{-1} \tag{A.42}$$

where,

$$S = (P\Lambda P^T) \tag{A.43}$$

such that P is the matrix of eigenvectors of $A^T A$ and,

$$\Lambda = \begin{bmatrix} \sqrt{\lambda_1} & 0 & 0 & 0 \\ 0 & \sqrt{\lambda_2} & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & 0 & \sqrt{\lambda_n} \end{bmatrix} \tag{A.44}$$

Appendix A

List of Software

Filename	Used in Section	Generated Figures	Function
GetTransMat_teats.m	General use.		Convert points from microscribe or camera coordinate frames to robot coordinate frame.
GetTransMat_cups.m	General use.		Convert points from microscribe coordinate frame to robot coordinate frame when the microscribe is situated to detect the milking cups.
Simulation01008d.m	4.2.2	4.1 - 4.3	Simulation comparing system model to experimental results.
Simulation01008b.m	4.4.2	4.4 - 4.6	Kalman filter simulation.
Simulation01014a.m	4.5.2	4.11 - 4.13	Particle filter simulation.
Simulation01014aa.m	4.5.2	4.14	Particle filter simulation - using experimental data.
Simulation01023.m	4.6.3.2	4.15 - 4.16	Orthogonalisation algorithm simulation.
Simulation01025.m	4.6.4.2	4.17	Mapping algorithm simulation.
ortho.m			Orthogonalisation algorithm called as a function by Simulation01023.m and Simulation01027.m.
Kalman_i.m	6.4.3	6.1	Variation of Kalman filter (i) applied to experimental data.
Kalman_ii.m	6.4.3	6.11	Variation of Kalman filter (ii) applied to experimental data.
Kalman_iii.m	6.4.3	6.12	Variation of Kalman filter (iii) applied to experimental data.
Kalman_iv.m	6.4.3	6.13	Variation of Kalman filter (iv) applied to experimental data.
Simulation01027.m	6.5	6.15 - 6.16	Compare orthogonalisation and mapping algorithms using experimental data.
test5.mat			Experimental data collected for use with Simulation01027.m
test3big.mat			Experimental data collected during 100-move test.

Table A.1: Matlab files.

Filename	Function	Called By
Coordinate frame setup.vi	Moves robot to a set of points in order to define microscribe coordinate frame.	
Grab and Snap 002.vi	Captures and stores images from cameras.	
Get Teat Coords - 003.vi	Top VI that selects teats from camera feeds, retrieves cups and places them on the teats.	
Get Teat Coords - 004.vi	Top VI that moves robot towards teats in 6 increments. Used for Data acquisition for Kalman filtering.	
Get Teat Coords - 005.vi	Top VI that moves robot towards teats in 100 increments with Kalman filter integrated.	
Get Teat Coords - 006.vi	Applies misidentification algorithms.	
Transform Camera to Robot.vi	Convert teat coordinates from camera frame to robot frame.	Get Teat Coords - 003.vi
Triangulate Teats - 001.vi	Perform triangulation for all four 3D points.	Get Teat Coords - 003.vi
Triangulation 002.vi	Find 3D coordinates from pixel values.	Triangulate Teats - 001.vi
The Cowminator - 004.vi	Retrieve and place cups.	Get Teat Coords - 003.vi
Mothership 011.vi	Place cups.	The Cowminator - 004.vi
Get Cups - 008.vi	Retrieve cups.	The Cowminator - 004.vi
Initialise All 002.vi	Return actuation system to initial positions.	The Cowminator - 004.vi
AX Control Cycle - SAFE.vi	Move robot at high or low speed.	The Cowminator - 004.vi; Mothership - 011.vi; Get Cups - 008.vi

Table A.2: Labview files - part 1.

Filename	Function	Called By
AX Control Cycle - fast.vi	Move robot at high speed.	AX Control Cycle - SAFE.vi
AX Control Cycle - slow.vi	Move robot at low speed.	AX Control Cycle - SAFE.vi
Adjust for phi 003.vi	Reposition toolplate to allow for change of angle phi.	Mothership - 011.vi; Get Cups - 008.vi
Calc Angle 002.vi	Find angle phi.	Mothership - 011.vi; Get Cups - 008.vi
Find Average Coordinates 002.vi	Find the centre of the measured teat cluster.	Mothership - 011.vi; Get Cups - 008.vi
Robot to Flange coord conversion - 002.vi	Find positions of teats realtive to centre of toolplate.	Mothership - 011.vi; Get Cups - 008.vi
X-Y cup control 006.vi	Control end-effector to position cups.	Mothership - 011.vi; Get Cups - 008.vi
Open Hand 003a.vi	Control end-effector to reach for or release cups.	Get Cups - 008.vi
EXTERNAL MOTORS ON.vi	Power on robot motors from external command.	The Cowminator - 004.vi; Initialise All - 002.vi
EXTERNAL MOTORS OFF.vi	Power off robot motors from external command.	The Cowminator - 004.vi; Initialise All - 002.vi
Initialise Linear.vi	Move linear axis to home position.	Initialise All - 002.vi
Move Linear.vi	Move linear axis to specified position.	Initialise All - 002.vi; X-Y cup control - 006.vi; Open Hand - 003a.vi
Revolute Control.vi	Move revolute axis to specified position.	Initialise All - 002.vi; X-Y cup control - 006.vi; Open Hand - 003a.vi
Get Theta.vi	Apply inverse kinematics to find required revolute angle.	X-Y cup control - 006.vi; Open Hand - 003a.vi

Table A.3: Labview files - part 2.

Filename	Function	Called By
Get Xl - 003.vi	Apply inverse kinematics to find required linear displacement for rear arms.	X-Y cup control - 006.vi; Open Hand - 003a.vi
Get Xl_forward 003.vi	Apply inverse kinematics to find required linear displacement for front arms.	X-Y cup control - 006.vi; Open Hand - 003a.vi
EXTERNAL PLAY START.vi	AX controller signalling.	AX Control Cycle fast.vi; AX Control Cycle - slow.vi
POSITION CHANGE SREQ PROMPT.vi	AX controller signalling.	AX Control Cycle fast.vi; AX Control Cycle - slow.vi
PROGRAM SELECT .vi	AX controller signalling.	AX Control Cycle fast.vi; AX Control Cycle - slow.vi
GENERIC READ 7344 PORT.vi	Read digital I/O signal on NI PCI-7344 card.	AX Control Cycle fast.vi; AX Control Cycle - slow.vi
GENERIC SET 7344 PORT.vi	Set digital I/O signal on NI PCI-7344 card.	AX Control Cycle fast.vi; AX Control Cycle - slow.vi
Linear Axis INIT.vi	Initialise NI PCI-7344 card.	Initialise Linear.vi
Linear Axis Home.vi	Move stepper motor until home switch is found.	Initialise Linear.vi
Linear Axis Move.vi	Move stepper motor to specified position.	Move Linear.vi
EPOS INIT_COM4only.vi	Open communications with servo controller	Revolute Control.vi
EPOS HOME.vi	Move servo motor until home switch is found.	Revolute Control.vi
EPOS MOVE.vi	Move servo motor to specified position.	Revolute Control.vi
EPOS CLOSE_COM4only.vi	Close communications with servo controller	Revolute Control.vi

Table A.4: Labview files - part 3.

Bibliography

[1] Department of Agriculture, Food and the Marine, "Annual report 2010." http://www.agriculture.gov.ie/media/migration/publications/2011/AnnualReport2010DAFF041011.pdf. Accessed November 20, 2012.

[2] Department of Agriculture, Food and the Marine, "Fact Sheet on Irish Agriculture 2011." http://www.agriculture.gov.ie/media/migration/publications/2011/FactSheetIrishAgJune11update.pdf. Accessed November 20, 2012.

[3] P. Dillon and C. O'Donoghue, "Providing technologies for profitable expansion," *TResearch*, vol. 4, no. 2, p. 22, 2009. http://www.teagasc.ie/publications/2009/15/15_tresearch200905.pdf.

[4] T. Donnellan, T. Hennessy, M. Keane, and F. Thorne, "Study of the International Competitiveness of the Irish Dairy Sector at Farm Level," tech. rep., Teagasc, 2011. http://www.teagasc.ie/publications/view_publication.aspx?PublicationID=1004, Accessed November 20, 2012.

[5] B. O'Brien, "Profiling the working year on irish dairy farms - identification of some work areas towards improvement in efficiency," *Irish Grassland Association Journal*, vol. 35, pp. 128–140, 2001.

[6] Central Statistics Office Ireland, "Regional population projections." http://www.cso.ie/en/media/csoie/releasespublications/documents/population/current/poppro.pdf. Accessed November 20, 2012.

[7] A. J. Bramley, *Machine Milking and Lactation*. Vermont: Insight Books, 1992.

[8] C. Culpin, *Farm Machinery*. London: Granada Publishing, 10 ed., 1992.

[9] J. White, "Design of a robotic manipulator for automatic application of milking cups," Master's Thesis, Dublin City University, 2006.

[10] J. Whipp, "Design and performance of milking parlours," in *Machine Milking and Lactation* (F. H. Dodd, G. A. Mein, and J. A. Bramley, eds.), pp. 285–310, Berkshire: Insight Books, 1992.

[11] SAC Netherlands BV., "RDS FutureLine MAX." `http://www.sac.eu/en/products/milking-cows/automatic-milking/rds-futureline-max`. Accessed November 20, 2012.

[12] DeLaval International AB., "DeLaval AMR." `http://www.delaval.com/en/About-DeLaval/Innovation-at-DeLaval/`. Accessed November 20, 2012.

[13] GEA Farm Technologies GmbH., "MIone The Milking Robot for Future Oriented Dairy Farms." `http://www.geagroup.com/en/produktnews/20090504-00324.html`. Accessed November 20, 2012.

[14] Lely Holding S.à.r.l., "Astronaut A4." `http://www.lely.com/en/milking/robotic-milkingsystem/astronaut-a4/lely-astronaut-a4#tab`. Accessed November 20, 2012.

[15] Fullwood Ltd., "Automated Milking." `http://www.fullwood.com/t/automatic-milking-system`. Accessed November 20, 2012.

[16] DeLaval International AB., "AMR Sytem Description." `http://www.delaval.com/en/About-DeLaval/Innovation-at-DeLaval/AMR-System-Overview/`. Accessed November 20, 2012.

[17] M. Kuczaj, W. Kruszyński, E. Pawlina, and J. Akińcza, "Relations between milk performance and udder dimensions of black-white cows imported from holland," *Electronic Journal of Polish Agricultural Universities*, vol. 3, no. 2, 2000. http://www.ejpau.media.pl/volume3/issue2/animal/art-01.html.

[18] D. G. Sorrenti, M. Cattaneo, and V. Villa, "Ultrasonic-based localization of cow teats for robotized milking," *Cybernetics and Systems*, vol. 39, no. 4, 2008.

[19] A. Ben Azouz, "Development of a teat sensing system for automated milking," Master's Thesis, Dublin City University, 2009.

[20] H. I. Christensen and G. D. Hager, "Sensing and Estimation," in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), ch. 4, Berlin: Springer, 2008.

[21] S. Thrun, W. Burgard, and D. Fox, "Bayes Filters," in *Probabilistic Robotics*, ch. 2.4, MIT Press, 2006.

[22] H. Durrant-Whyte and T. C. Henderson, "Bayes' Rule," in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), ch. 25.1.1, Berlin: Springer, 2008.

[23] P. Swerling, "A proposed stagewise differential correction procedure for satellite tracking and prediction," Technical Report P-1292, RAND Corporation, 1958.

[24] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transactions of the ASME–Journal of Basic Engineering*, vol. 82, no. Series D, pp. 35–45, 1960.

[25] S. Thrun and J. J. Leonard, "Simultaneous Localisation and Mapping," in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), ch. 37, Berlin: Springer, 2008.

[26] W. Bell and S. Hillmer, "Initializing the Kalman filter for nonstationary time series models," *Journal of Time Series Analysis*, vol. 12, no. 4, pp. 283–300, 1991.

[27] Y. Bentz, L. Boone, and J. Connor, "Modelling stock return sensitivities to economic factors with the kalman filter and neural networks," in *Proceedings of the IEEE/IAFE Conference on Computational Intelligence for Financial Engineering*, pp. 79 –82, March 1996.

[28] P. J. Sherman, T. Jnsson, and H. Madsen, "A kalman filter based dsp method for prediction of seasonal financial time series with application to energy spot price prediction," in *Statistical Signal Processing Workshop (SSP), 2011 IEEE*, pp. 33 –36, June 2011.

[29] M. S. Grewal and A. P. Andrews, *Kalman Filtering Theory and Practice Using Matlab*, p. 4. Wiley, 2001.

[30] N. Gordon, D. Salmond, and A. Smith, "Novel approach to nonlinear/non-gaussian bayesian state estimation," *Radar and Signal Processing, IEE Proceedings F*, vol. 140, pp. 107 –113, April 1993.

[31] N. Metropolis and S. Ulam, "The Monte Carlo Method," *Journal of the American Statistical Association*, vol. 44, no. 247, pp. 335–341, 1949.

[32] S. Thrun, W. Burgard, and D. Fox, "The Particle Filter," in *Probabilistic Robotics*, ch. 4.3, MIT Press, 2006.

[33] H. Durrant-Whyte and T. C. Henderson, "Sequential Monte Carlo Methods," in *Springer Handbook of Robotics* (B. Siciliano and O. Khatib, eds.), ch. 25.1.4, Berlin: Springer, 2008.

[34] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter*. Artech House, 2004.

[35] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 174–188, 2002.

[36] J. D. Hol, T. B. Schön, and F. Gustafsson, "On resampling algorithms for particle filters," in *Nonlinear Statistical Signal Processing Workshop*, 2006.

[37] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2003.

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

[39] J.-Y. Bouguet, "Camera calibration toolbox for matlab." `http://www.vision.caltech.edu/bouguetj/calib_doc/`. Accessed November 20, 2012.

[40] Janne Heikkilä and Olli Silvén, "A four-step camera calibration procedure with implicit image correction," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1106 –1112, June 1997.

[41] D. C. Brown, "Close range camera calibration," *Photogrammetric Engineering*, vol. 37, no. 8, pp. 855–866, 1971.

[42] P. J. McKerrow, *Introduction to Robotics*. Addison-Wesley, 1998.

[43] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, p. 154. Cambridge University Press, 2003.

[44] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, p. 157. Cambridge University Press, 2003.

[45] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, pp. 153–157. Cambridge University Press, 2003.

[46] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, p. 184. Cambridge University Press, 2003.

[47] O. Faugeras, *Three Dimensional Computer Vision A Geometric Viewpoint*, p. 166. MIT Press, 1993.

[48] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, pp. 241–245. Cambridge University Press, 2003.

[49] A. Hunt Duffy, "Teat detection for an automated milking system," Master's Thesis, Dublin City University, 2006.

[50] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, p. 179. Cambridge University Press, 2003.

[51] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, p. 263. Cambridge University Press, 2003.

[52] "OpenCV Wiki." `http://opencv.willowgarage.com`. Accessed November 20, 2012.

[53] "GML C++ Camera Calibration Toolbox." `http://graphics.cs.msu.ru/en/science/research/calibration/cpp`. Accessed November 20, 2012.

[54] D. Simon, "Kalman filtering," *Embedded Systems Programming*, vol. 14, no. 6, pp. 72–79, 2001. Available online at `http://academic.csuohio.edu/simond/courses/eec644/kalman.pdf`, Accessed November 20, 2012.

[55] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter*, p. 5. Artech House, 2004.

[56] G. Welch and G. Bishop, "An introduction to the Kalman filter," Technical Report TR 95-041, Department of Computer Science University of North Carolina at Chapel Hill, 2006.

[57] B. Ristic, S. Arulampalam, and N. Gordon, *Beyond the Kalman Filter*, p. 8. Artech House, 2004.

[58] S. Thrun, W. Burgard, and D. Fox, "The Kalman Filter," in *Probabilistic Robotics*, ch. 3.2, MIT Press, 2006.

[59] S. G. Tzafestas, "Kalman Filter Implementation," in *Microprocessors in Robotic and Manufacturing Systems*, ch. 16.4, London: Springer, 1991. Available online at `http://bit.ly/uhUFZg`, Accessed November 20, 2012.

[60] C. Newland and D. Gray, "Time invariant steady-state kalman filter for image super-resolution," in *Proceedings of Image and Vision Computing New Zealand*, (Dunedin, New Zealand), IVCNZ, 2005.

[61] E. Walter and L. Pronzato, *Identification of Parametric Models from Experimental Data*, ch. 4.1.6.8. Springer, 1997.

[62] B. Ristic, S. Arulampalam, and N. Gordon, "Monte Carlo Integration," in *Beyond the Kalman Filter*, Artech House, 2004.

[63] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics*, p. 112. MIT Press, 2006.

[64] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte Carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation (ICRA99)*, May 1999.

[65] J.-S. Gutmman and D. Fox, "An experimental comparison of localization methods continued," in *Proceedings of the 2002 IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, (Lausanne, Switzerland), October 2002.

[66] S. Thrun, W. Burgard, and D. Fox, "Random Particle MCL: Recovery from Failures," in *Probabilistic Robotics*, ch. 8.3.5, MIT Press, 2006.

[67] A. Kong, J. S. Liu, and W. H. Wong, "Sequential imputations and bayesian missing data problems," *Journal of the American Statistical Association*, vol. 89, no. 425, pp. 278–288, 1994.

[68] S. Lenser and M. Veloso, "Sensor resetting localization for poorly modelled mobile robots," in *Proceedings of the 2000 IEEE International Conference on Robotics & Automation*, (San Fransisco, CA), April 2000.

[69] S. Thrun, D. Fox, W. Burgard, and F. Dellaert, "Robust Monte Carlo localization for mobile robots," *Artificial Intelligence*, vol. 128, no. 1-2, pp. 99–141, 2000.

[70] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, 1981.

[71] Nachi Fujikoshi Corp., "Nachi SC35F/SC50F Catalog." `http://www.nachi-fujikoshi.co.jp/eng/web/pdf/7307-7.pdf`. Accessed November 20, 2012.

[72] E. Walter and L. Pronzato, *Identification of Parametric Models from Experimental Data*, p. 106. Springer, 1997.