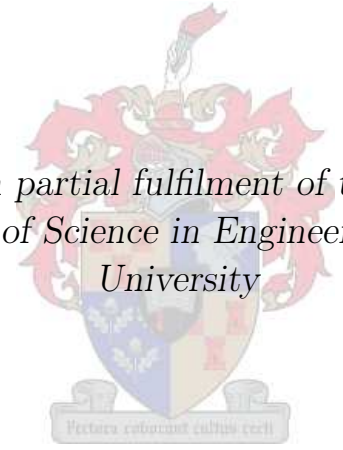
# Visual Servo Control for a Human-Following Robot

by

Michael Glen Burke



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

Supervisors:

| | | |
|---|---|---|
| Mr JAA Engelbrecht | Dr W Brink | Prof. K Schreve |
| Dept. E&E Engineering | Dept. Mathematical Sciences | Dept. M&M Engineering |

March 2011

# Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the owner of the copyright thereof (unless to the extent explicitly otherwise stated) and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: March 2011 .....

# Abstract

This thesis presents work completed on the design of control and vision components for use in a monocular vision-based human-following robot. The use of vision in a controller feedback loop is referred to as vision-based or visual servo control. Typically, visual servo techniques can be categorised into image-based visual servoing and position-based visual servoing. This thesis discusses each of these approaches, and argues that a position-based visual servo control approach is more suited to human following.

A position-based visual servo strategy consists of three distinct phases: target recognition, target pose estimation and controller calculations. The thesis discusses approaches to each of these phases in detail, and presents a complete, functioning system combining these approaches for the purposes of human following.

Traditional approaches to human following typically involve a controller that causes platforms to navigate directly towards targets, but this work argues that better following performance can be obtained through the use of a controller that incorporates target orientation information. Although a purely direction-based controller, aiming to minimise both orientation and translation errors, suffers from various limitations, this thesis shows that a hybrid, gain-scheduling combination of two traditional controllers offers better target-following performance than its components.

In the case of human following the inclusion of target orientation information requires that a definition and means of estimating a human's orientation be available. This work presents a human orientation measure and experimental results to show that it is suitable for the purposes of wheeled platform control. Results of human following using the proposed hybrid, gain-scheduling controller incorporating this measure are presented to confirm this.

# Uittreksel

Die ontwerp van 'n visiestelsel en beheer-komponente van 'n enkel-kamera robot vir die volging van mense word hier aangebied. Die gebruik van visuele terugvoer in die beheerlus word visie-gebaseerde of visuele servobeheer genoem. Visuele servobeheer tegnieke kan tipies onderskei word tussen beeld-gebaseerde servobeheer en posisie-gebaseerde visuele servobeheer. Altwee benaderings word hier bespreek. Die posisie-gebaseerde benadering word aanbeveel vir die volging van mense.

Die posisie-gebaseerde servobeheertegniek bestaan uit drie duidelike fases: teiken herkenning, teiken oriëntasie bepaling en die beheerder berekeninge. Benaderings tot elk van hierdie fases word hier in detail bespreek. Dan word 'n volledige funksionele stelsel aangebied wat hierdie fases saamvoeg sodat mense gevolg kan word.

Meer tradisionele benaderings tot die volging van mense gebruik tipies 'n beheerder wat die platvorm direk laat navigeer na die teikens, maar hier word geargumenteer dat beter werkverrigting verkry kan word deur 'n beheerder wat die teiken oriëntasie inligting ook gebruik. 'n Suiwer rigting-gebaseerde beheerder, wat beide oriëntasie en translasie foute minimeer, is onderhewig aan verskeie beperkings. Hier word egter aangetoon dat 'n hibriede, aanwinsskedulerende kombinasie van die twee tradisionele beheerders beter teikenvolging werkverrigting bied as die onderliggende twee tegnieke.

In die geval van die volging van mense vereis die insluiting van teiken oriëntasie inligting dat 'n definisie van die persoon se oriëntasie beskikbaar is en dat dit geskat kan word. 'n Oriëntasie maatstaf vir mense word hier aangebied en dit word eksperimenteel getoon dat dit geskik is om 'n platvorm met wiele te beheer. Die resultate van die volging van mense wat die voorgestelde hibriede, aanwins-skedulerende beheerder gebruik, met hierdie maatstaf, word ter ondersteuning aangebied.

# Acknowledgements

I would like to express my sincere gratitude to the following people and organisations for their contributions to the project:

- My triumvirate of supervisors, Japie, Kristiaan and Willie, for their suggestions and guidance, and for gracefully enduring the 1285.33 km separating us for much of my studies (a special thank you to Willie, for all the time spent assisting via Google Talk);
- My family and friends, for their encouragement and support throughout this process – especially Lisa, for listening to my grievances at lunch each day;
- The Mobile Intelligent Autonomous Systems (MIAS) group at the Council for Scientific and Industrial Research (CSIR), who funded this work;
- All my fellow MIAS members, but above all Deon Sabatta and Jonathan Claassens for their keen interest in my work and valuable suggestions.

# Contents

# List of Figures

# List of Tables

# Nomenclature

**Abbreviations**

| | |
|---|---|
| 2D | Two Dimensional |
| 3D | Three Dimensional |
| CCD | Charge-coupled Device |
| CSIR | Council for Scientific and Industrial Research |
| DLT | Direct Linear Transform |
| DOF | Degree of Freedom |
| EKF | Extended Kalman Filter |
| FAST | Features from Accelerated Segment Test |
| GPU | Graphical Processing Unit |
| IBVS | Image-based Visual Servoing |
| ICP | Iterative Closest Point |
| KLT | Kanade Lucas Tracker |
| LIDAR | Light Detection and Ranging |
| MIAS | Mobile Intelligent Autonomous Systems |
| OpenCV | Open Computer Vision Library |
| PBVS | Position-based Visual Servoing |
| PCA | Principal Component Analysis |
| P$n$P | Perspective-$n$-point |
| POSIT | Pose from Orthography and Scaling with Iterations |
| PTU | Pan-Tilt Unit |
| RANSAC | Random Sample Consensus |
| RAM | Random Access Memory |
| RFID | Radio Frequency Identification |
| ROI | Region of Interest |
| SIFT | Scale Invariant Feature Transform |
| SIS-R | Sequential-importance-sampling Re-sampling |
| SURF | Speeded Up Robust Features |
| SVD | Singular Value Decomposition |

| | |
|---|---|
| UAV | Unmanned Aerial Vehicle |

**Units**

| | |
|---|---|
| m | meters |
| s | seconds |
| ms | milliseconds |
| m/s | meters per second (translational speed) |
| $rad$ | radians (measure of angle) |
| $rad/s$ | radians per seconds (angular speed) |
| $^\circ$ | degrees (measure of angle) |
| fps | frames per second (frame-rate measure) |
| GB | Gigabytes |
| GHz | Gigahertz |

**Camera Geometry Variables**

| | |
|---|---|
| $\mathbf{H}$ | $3 \times 3$ homography matrix |
| $\mathbf{K}$ | $3 \times 3$ camera calibration or intrinsic matrix |
| $f$ | camera focal length |
| $\mathbf{n}$ | $3 \times 1$ surface normal |
| $\mathbf{R}$ | $3 \times 3$ rotation or direction cosine matrix |
| $\mathbf{t}$ | $3 \times 1$ translation vector |
| $[\mathbf{R}\|\mathbf{t}]$ | concatenated $3 \times 4$ rotation and translation matrix |
| $\lambda$ | scaling factor |

**Kalman Filter Variables**

| | |
|---|---|
| $\hat{\mathbf{x}}_{k-1\|k-1}$ | previous state estimate |
| $\hat{\mathbf{x}}_{k\|k-1}$ | predicted state vector |
| $\mathbf{u}_k$ | vector of controls |
| $\mathbf{P}_{k\|k-1}$ | predicted covariance |
| $\mathbf{P}_{k-1\|k-1}$ | previous covariance estimate |
| $\mathbf{Q}_k$ | current process noise covariance |
| $\mathbf{z}_k$ | current measurement vector |
| $\tilde{\mathbf{y}}_k$ | current measurement residual |
| $\mathbf{S}_k$ | current covariance residual |
| $\mathbf{R}_k$ | measurement noise covariance |
| $\mathbf{K}_k$ | Kalman gain |
| $\hat{\mathbf{x}}_{k\|k}$ | current state estimate |

| | |
|---|---|
| $\mathbf{P}_{k\|k}$ | current covariance estimate |

**Controller Variables**

| | |
|---|---|
| $v$ | platform forward velocity |
| $\omega$ | platform rotational velocity |
| $t_x$ | horizontal translation component between two camera viewpoints, measured up to scale |
| $t_z$ | translation component between two camera viewpoints, measured up to scale along one of the camera's optical axes |
| $\alpha$ | pan angle of pan-tilt unit |
| $\phi$ | relative yaw orientation error between target and camera |

# Chapter 1

# Introduction

## 1.1 Background

The ability of a mobile robot to track and follow a moving target is required in a wide variety of applications, particularly when the coordination of multiple robots is required. Cooperative robotics requires that individual robots are aware of the positions and behaviours of surrounding agents prior to any useful collaborative action. One of the more common tasks required by cooperative agents is formation control, defined by Das *et al.* (2002) as the problem of controlling the relative positions and orientations of robots in a group, while allowing the group to move as a whole. Many different formations have been proposed, but the leader-follower formation is one of the most common.

The leader-follower formation control task requires that a mobile robot follows a target, maintaining a specified range and relative orientation. Such a task is particularly useful for robotic convoys, where a lead vehicle is teleoperated and a cascade of agents follow, allowing the transport of supplies or vehicles through dangerous areas, without the need to risk human lives. Leader-follower formation control is also envisaged to be of great use in intelligent transportation systems.

Although introduced in the context of cooperative robotics, the leader-follower formation control definition also applies to the case of uncooperative or unpredictable targets. Uncooperative targets range from elusive, manoeuvring objects to collaborating independent agents without the ability to communicate their intentions. A good example of the latter is that of a human-following robot.

Robots equipped with the ability to follow humans could prove particularly useful, especially within the service robotics industry. Robotic mules could follow humans out to a point and then move back and forth ferrying burdens. Another potential application is in search and rescue, where a robot follows teams of medics, and returns stabilised patients to field hospitals, with only a single medic required to walk alongside, leaving the others free to continue

work on other patients in a disaster area.

If these applications are to become a reality, human-following robots not only need to detect, recognise and track humans in real time, but also navigate towards them in an intelligent manner. This is particularly challenging, as humans are classed as uncooperative targets, since they lack the ability to communicate with robotic vehicles. As a result, an efficient means of perception and target recognition is crucial.

Human-following robots are typically equipped with a diverse and varying combination of sensors for locating and recognising targets. Light detection and ranging (LIDAR), for example, provides accurate bearing measurements but suffers from potential ambiguity in target recognition. Electronic tethering techniques that use radio frequency identification (RFID) are effective, but require that the human followed wear a tracking device and still need a secondary sensor for greater measurement accuracy. As a result many systems employ vision, selected for its ability to provide abundant information about the robot's environment passively, and at relatively high speeds.

The control of mobile robots using vision in the feedback loop falls into the well-studied field of visual servo control. At present two types of visual servoing strategies are popular, image-based visual servoing (IBVS) and position-based visual servoing (PBVS). IBVS refers to the control of a system using calculations performed in the image plane, by making use of image coordinates. PBVS defines control strategies in terms of the vision system's position relative to some reference coordinates in the observed world.

Various types of vision system or camera placement are used in conjunction with these visual servoing strategies. These include fixed location vision systems, pan-tilt and eye-in-hand configurations. The eye-in-hand configuration refers to a vision system fixed to the controlled system, where camera motion is constrained to movements made by the controlled system.

The primary application of visual servoing is to control the motion of an effector relative to a target. Two strategies of motion control are used, point-to-point positioning and pose or direction-based motion control, the primary difference being that target orientation is taken into account in direction-based motion control. The inclusion of target orientation information in the control strategy introduces numerous benefits, many of which will be addressed here.

## 1.2 Problem Statement

The problem addressed in this thesis is the design and implementation of a suitable visual servo control system that allows a wheeled mobile robot to track and follow a human. Although human following is the primary application in mind, efforts are made to maintain generality so as to ease the transition to arbitrary object following.

It is important to note that visual servo control is reactive and as a result

advanced path planning and navigation schemes fall beyond the scope of this work. In addition, no collision avoidance is considered in this thesis, due to the servoing nature of the controllers implemented. Extensions to control strategies that allow for collision avoidance are available, however, and could be used to supplement the human follower without affecting the conclusions made in this work.

System operation is restricted to indoor environments under relatively controlled lighting conditions. While constrained in size, indoor environments are potentially cluttered, which makes target detection difficult.

The use of visual information requires that targets be discriminable within image scenes. As a result, no camouflaged targets are considered and it is assumed that the targets followed are sufficiently salient.

The primary challenges this work aims to overcome are problems of perception and target detection in real time, relative orientation estimation, and the control of a commercially available platform with constrained dynamics. The restriction to only a monocular vision system introduces further difficulties with regard to 3D (three dimensional) reconstruction.

Although the platform used for the implementation of the human-following system is equipped with odometric sensors, the requirement that vision is the only means of perception and feedback is imposed and odometric information is discarded. This restriction allows the human-following system to be implemented on simpler, less costly vehicles, but adds to the challenges in implementation.

A requirement that the human-following robot be able to detect targets when travelling over uneven terrain is also imposed. This requirement arises from the need to detect the pose of an awkwardly moving human, which can be considered analogous to detecting a target when traversing uneven terrain.

A direction-based control strategy that incorporates target orientation has many potential benefits and would prove useful if applied to the human-following problem. Unfortunately, there is no clear definition of human orientation, but a direction-based control system for human-following requires that some measure of human orientation be available.

This work presents a visual servoing solution to the human-following problem. The benefits of obtaining target orientation are made clear, and a control approach that better exploits these benefits is introduced. A definition and means of estimating human orientation is presented and the benefits of a human-following strategy that includes orientation information are evaluated.

## 1.3 Methodology

A phased and modular approach is used to develop a solution to the human-following problem. Four primary components – recognition, pose estimation, tracking and control – were identified as critical to the problem solution after

analysis of visual servo control literature and background work. Recognition refers to the detection of the human to be followed. After recognition, pose estimation extracts the position and attitude of the target, measured relative to the camera. The position and attitude of the target are then refined through tracking. A tracking algorithm also allows for position and attitude to be estimated during brief time periods when no target measurements are made. Finally, the refined estimates of target information are used by a control system to generate motion commands for a wheeled platform.

A common procedure is followed in the design of each of these components. Initially, various algorithms are identified and compared, based on a thorough investigation of literature published on the topic. Critical criteria considered necessary for each component are specified and used to select those approaches most suitable to the problem. These methods are implemented and tested through practical experimentation and simulations, before being reduced to a single solution through verification based on the critical criteria. The critical criteria predominantly focus on aspects of speed and accuracy, but also consider the limitations of the available hardware, such as platform dynamics and vision system constraints.

The benefits of each approach are considered and in many cases, changes are made to adapt the techniques to the target-following problem. Further testing of the approaches occurs once integration is complete.

At first, all work is geared towards developing both a direction-based and point-to-point generalised target-following system and a comparison of the two is made. Thereafter, attempts at expanding and adapting these approaches are undertaken, in order to meet the more specific challenge of human-following.

This adaptation is relatively simple in the case of point-to-point target following, but requires that a measure of human orientation be available for pose-based following. A definition and measure of human orientation is introduced and validated through practical experimentation.

Further comparisons are made within the human-following domain, in order to determine whether the benefits identified in the generalised case hold for the more specific human-following scenario. Finally, the overall human-following system is validated through experimental testing in various scenarios designed to test the abilities of the system.

## 1.4 Aims and Objectives

The primary objectives of this research project are:

- to investigate various object recognition and pose estimation algorithms and select those most suited to real-time control applications with human following in mind;

- to develop an effective mobile object tracking system, which can be implemented on typical differential drive[1] mobile platforms;
- to analyse the performance of selected object recognition, pose estimation and control algorithms;
- to identify the benefits of both direction-based controllers and point-to-point controllers;
- to adapt these controllers to better suit a following application;
- to design and implement a means of determining human orientation, and evaluate the benefits of a control strategy that utilises it;
- to suitably modify existing mobile platforms so that a leader-follower tracking configuration can be evaluated therewith.

The end goal of the research is to develop a complete robotic system, which makes use of dynamic visual servoing on-board a mobile robot, to track and follow a human leader. All research conducted will be geared towards fulfilling this objective, but emphasis will be placed on improving the traditional human-following approach by including orientation information. In doing so three research questions will be answered.

- Is there any benefit in direction-based control over point-to-point control for a generic target follower?
- If the benefits are negligible, is it possible to use orientation information in a modified control scheme in such a way as to enhance them?
- If so, is there a measure of human pose or orientation that makes it possible to incorporate these benefits into a human-following system?

The use of a vision-based object recognition and pose estimation system that extracts human pose information will be established if these questions are answered.

## 1.5 Contribution

Although the primary goal of this work is to design and implement a visual servo control system for a human-following robot, novel contributions to the field of robotics are made. The primary contributions of this work are listed below.

[1] A differential drive platform is one where the turning speed of the platform is determined by the speed difference between two wheels, positioned on opposite sides of the platform.

- A generalised target tracking and following system is designed and implemented.
- A hybrid control approach combining the benefits of direction-based motion control and point-to-point controllers is designed.
- A definition of human orientation is introduced, together with a means of measuring it.
- A human-following system incorporating human orientation is implemented and compared with the traditional human-following approach using point-to-point controllers.

The definition and estimation of human orientation is applicable elsewhere in field robotics, despite being defined in the context of visual servo control. Planner-based navigation systems could make use of orientation for human following, while human robot interaction studies would benefit from the additional information supplied.

A complete and functioning human-following system could also be used as the building block in numerous applications, such as the search and rescue scenario described earlier, where mobile platforms act as human assistants in disaster areas.

## 1.6 Thesis Outline

This thesis discusses the use of visual servo control for a human-following robot. An extensive literature study introducing the components required for a human-following robot, together with previous human-following work, is presented in Chapter 2. In Chapter 3, a theoretical analysis and design of the proposed solution is undertaken. Each of these chapters contains subsections relating to the four primary system components (recognition, pose estimation, tracking and control).

Simulated and experimental results verifying and validating the various system components are presented in Chapter 4, together with an analysis of the system limitations. Finally, Chapter 5 provides conclusions and recommendations for future approaches, based on the findings of this study.

# Chapter 2

# Literature Study

In this chapter, literature related to visual servo control and the target following problem is discussed. As previously mentioned, the goal of this work is to utilise visual servo control on-board a wheeled mobile platform to follow a human target. This requires that a target object is detected and a measure of its pose relative to some desired configuration made, with motion commands generated in order to correct any differences between the observed configuration and the desired one.

Initially, the two primary variants of visual servo control are introduced, followed by an introduction to object recognition strategies. The components of a position-based visual servo control system – pose estimation, tracking and motion control – are discussed thereafter. Finally, work directly relating to the human-following problem is reviewed.

## 2.1 Visual Servo Control

The primary objective in visual servoing is to minimise the error between a desired state and the current state, which is determined through image measurements in conjunction with other relevant information such as camera calibration and motion compensation parameters. Put simply, visual servo control aims to generate motion commands so that the current scene viewed resembles an image of the desired scene.

The two main visual servoing approaches, Image-based Visual Servoing (IBVS) and Position-based Visual Servoing (PBVS), are discussed in detail in the tutorial by Hutchinson *et al.* (1996) and in the work of Chaumette and Hutchinson (2008), which provides a review of visual servoing and tracking trends.

PBVS methods are more intuitive as they operate in three distinct stages. Initially, a target object is detected and recognised in the image. Thereafter, the position and orientation of the target object is determined through the process of pose estimation. Finally, a control algorithm compares these pose

measurements to those desired and generates motion commands that cause the platform to minimise this difference.

IBVS approaches, on the other hand, discard the pose estimation stage of PBVS. Here, a control algorithm generates motion commands based directly on errors between the locations of detected features[1] and their desired positions. IBVS methods are desirable due to the computational speed gained by performing all calculations in the image plane, but difficulties in extracting information regarding rotational motion and the complex control strategies required make them unpopular. IBVS strategies can also potentially suffer from stability issues as they are vulnerable to task singularities and can fail to converge due to the presence of local minima (Chaumette (1998)).

IBVS methods can be designed to ensure that stationary targets are kept in view, as they perform control in the image plane. Unfortunately, this can lead to non-ideal camera trajectories. In contrast, PBVS methods allow for the decoupling of rotational and translational motion, and hence simplify the control problems associated therewith. Control is performed based on physical parameters, which results in superior camera trajectories. Unfortunately, as control is removed from the image plane, it is no longer possible to ensure that the target object remains visible over the resultant trajectory.

In general though, the controllers used in PBVS present attractive stability properties. Unfortunately, PBVS approaches require 3D object models and that 3D localisation problems be solved in real time. Chaumette (1998) notes that the stability of PBVS techniques is largely determined by the efficacy of the pose estimation algorithm used.

More advanced approaches to the visual servoing problem are also examined in the work of Chaumette and Hutchinson (2008) and include hybrid, 2.5D or partitioned visual servoing approaches, which attempt to combine the best features of IBVS and PBVS methods. 2.5D visual servoing, first proposed by Malis *et al.* (1999), does not require a 3D model of the object and still allows for a decoupled control law, but is more sensitive to image noise.

## 2.2 Object Recognition

The discussion of visual servo control strategies assumes that a target object has already been detected and recognised in the image scene. Unfortunately, object recognition is an extremely challenging topic in the field of computer vision and one of the primary difficulties encountered when designing practical visual servo control systems.

Hutchinson *et al.* (1996) note that there is a tendency to simplify the object recognition stages of visual servoing, by making the objects clearly discernible, so that more attention can be placed on the controller side of visual servoing.

[1]Image features or keypoints are salient points in an image, typically located at corners, edges and areas of sharp contrast.

Various camera strategies have also been used to simplify the visual servoing process. An omnidirectional vision system was used by Das *et al.* (2001) to ease the recognition and pose determination process. Their vision system made use of a catadioptric sensor to provide a bird's-eye-view of the area around a robot.

Appearance-based object recognition techniques such as colour histogram matching (Gevers and Smeulders (1999)) are often used in visual servo control applications because of their ease of implementation. Unfortunately, colour matching leads to difficulties in extracting reliably positioned feature points and is subject to mismatches. The use of fiduciary markers alleviates potential mismatches, but requires interference with the target, which is not always possible. One of the most common fiduciary approaches, template matching, involves the design of an easily recognised marker and complementary filter, which is then correlated with the image scene. Unfortunately, this approach is often unable to recognise objects when they are viewed from distinctly different viewpoints.

Geometric object recognition techniques are popular alternatives to appearance-based approaches. Here, recognition is based on the object shape, as opposed to its appearance. These techniques rely on edge detection (Torre and Poggio (1984)) and geometric information of the target for recognition. Kass *et al.* (1988) propose the use of a snake or energy minimising spline that pulls in towards features such as lines and edges, given certain geometric constraints. Active contour models such as these can prove effective, but are quite slow and prone to error in extremely cluttered images. Snakes have been used successfully for leader-follower formation control, however, in the work of Vela *et al.* (2009). Here, active contours were used by an Unmanned Aerial Vehicle (UAV) to track and follow a leader.

Daniilidis and Eklundh (2008) provide a review of 3D vision and recognition techniques most commonly used in field robotics. Here, recognition strategies generally rely on appearance-based methods and the extraction of distinct features. Feature-based approaches generally operate in two stages. Initially, distinct and key features are detected within the scene. A descriptor of the region around each feature is then created. Descriptors are compared and an object is recognised if sufficient matched descriptors are found.

### 2.2.1 Feature Detectors

A vast amount of literature dedicated to the comparison of feature detectors is available. This section presents a brief overview of feature detectors that form the basis of those used in the feature matching schemes considered in this work.

Ideally, features used should be distinctive, easily recognised, and invariant to changes in both lighting and observer motion (Daniilidis and Eklundh (2008)). Analysis of the work by Schmid *et al.* (1998), Trajkovic and Hedley

(1998) and Zheng *et al.* (1999) shows that the following parameters are of interest when quantifying feature detector performance:

**detection rate:** the percentage of correctly detected features;

**localisation:** the accuracy of detected feature position;

**repeatability rate:** a measure of how frequently the same feature is detected in a scene over varying viewing conditions;

**robustness to noise:** the ability of features to be detected, despite the addition of noise to an image;

**speed:** the time taken for features to be extracted from an image.

One of the earliest feature detectors proposed was the Morevec corner detector (Morevec (1977)). This detector looks for features with extremely low self-similarity, or patches around features that differ from the feature surroundings. Unfortunately this feature detector is anisotropic[2] as it only measures similarity in discrete directions. This means that the detection of features is not rotationally invariant and has poor repeatability.

The Harris or Plessey edge and corner detector (Harris and Stephens (1988)) is an expansion of the Morevec corner detector with better repeatability and a higher detection rate. The Harris operator is isotropic as it searches for self similarity in all directions. In addition, the Harris operator results in a less noisy response than the Morevec detector, as it makes use of a circular Gaussian window function to test for self similarity. The use of this Gaussian also allows for adaptations to provide scale invariance.

The concept of scale in an image is easily understood. As objects in an image are viewed from further away, the level of visible detail is reduced. This reduction in detail can be imitated by a scale-space representation of an image, as discussed in Appendix A.3.

The multi-scale Harris operator is extremely popular within the computer vision community and forms the basis for a large variety of feature detectors. Noble (1989) proposed a filtered keypoint selection criteria for the Harris operator that results in better keypoint localisation. Unfortunately, neither this approach nor the original Harris operator is affine[3] invariant.

Mikolajczyk and Schmid (2002) proposed a modification to the multi-scale Harris operator that provides affine invariance through the use of an affine

[2]An anisotropic feature detector is one where the response is direction dependent. This is not ideal, as differing responses in various directions mean that the detector is not invariant to changes in rotation. An isotropic detector, which produces the same response over all directions, is preferable as it results in rotational invariance.

[3]An affine transformation consists of a linear transformation followed by a translation. The linear transformation could be a rotation, scaling or shear.

Gaussian scale-space. This scale space is generated by convolution with non-uniform Gaussian kernels which simulates heterogeneous scale change in different directions.

Although the modified Harris operators are able to detect good features that can be found over a variety of viewpoint changes, they do so at the expense of computation time. This drawback has led to an alternative set of feature detection techniques. Instead of dedicating time to finding an optimum set of features that can be detected, and hence matched, from a variety of viewpoints when using a generic matching scheme, these approaches aim to find less ideal features suited to a particular matching or tracking technique.

Features from accelerated segment test (FAST), proposed by Rosten and Drummond (2005), is a method of this type. The FAST detector identifies keypoints where the intensity of pixels on the edge of a Bresenham circle[4] differs greatly from that of the circle centre. The approach is made extremely fast by optimising the order in which pixels are tested. Unfortunately, this technique sacrifices feature quality for speed and requires an extremely robust matching scheme.

Shi and Tomasi (1994) introduced the concept of a feature selection criterion that is based on the operation of the tracking scheme they used to match features. Their selection criterion is closely related to the Harris operator, but a better measure when tracking affine feature motions.

Figure 2.1 shows results of the Harris, Harris-Noble and Shi-Tomasi feature detectors on a sample image undergoing various transformations. The number of features detected decreases progressively with each detector as the respective selection criterion is taken into account. The greyscale colour space used by the figure provides a clue as to the extraction of feature locations. Both positive maxima (white) and negative minima (black) are used as features in the traditional Harris operator. The Harris-Noble operator causes features to emerge as maxima (white), on a positive scale, while the Shi-Tomasi operator results in features represented by minima (black) on a negative scale.

The use of features or keypoints may not be ideal from an object recognition perspective. Often, objects do not have single interest points, but larger salient regions (blobs) instead. In such situations, feature-based recognition strategies could fail. Various blob detectors have been introduced in an attempt to remedy this. Unfortunately, blob detectors generally lack the localisation performance of feature detectors and as a result are not suited to visual servo control applications.

It is clear that there are numerous feature detection schemes available, each with varying properties. However, the choice of feature detector does not impact object recognition as significantly as the descriptor or matching scheme used. The feature detector is merely responsible for selecting points

[4]A Bresenham circle refers to the midpoint circle algorithm used to draw the points of a circle in computer graphics.

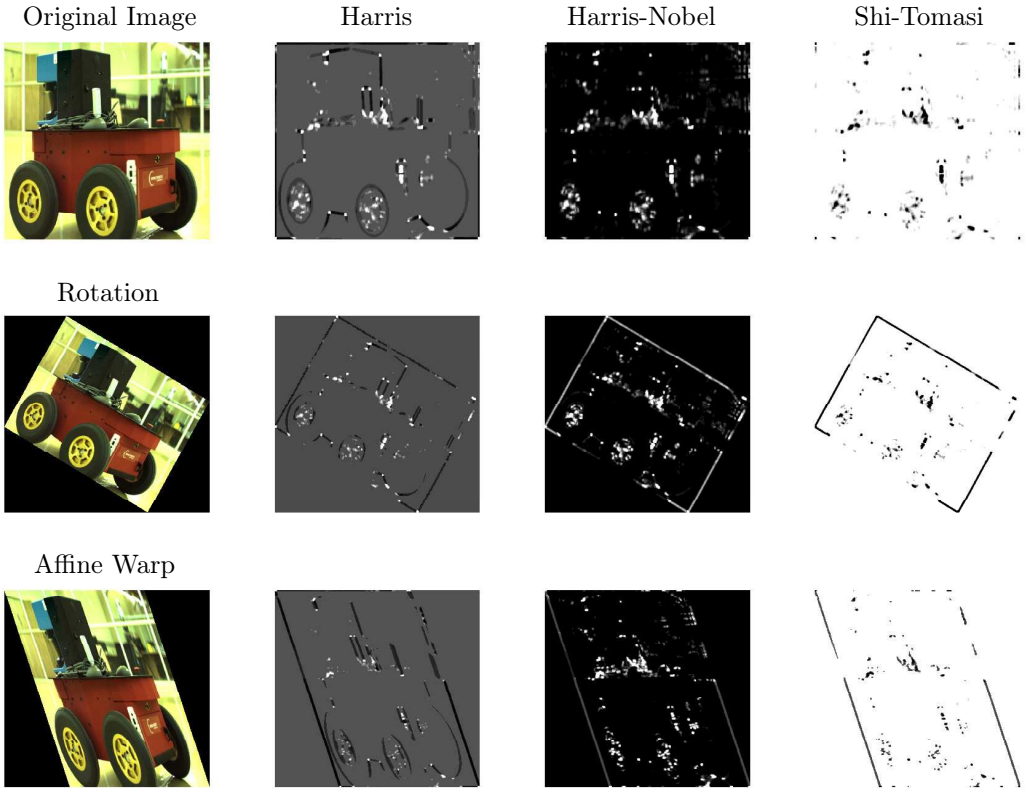



Figure 2.1: Results of the Harris, Harris-Nobel and Shi-Tomasi operators acting on images subjected to rotational and affine transformations.

of interest in an image, while the matching scheme needs to extract unique information of the areas about keypoints and then associate this with other areas containing similar information. In fact, in the author's opinion, almost any modern feature detector is suitable for use in a matching scheme, and increasingly complex detection schemes result in only minor improvements in matching performance. This is due to the high repeatability of most modern feature detection algorithms, which implies that sufficient features of similar types are likely to be detected across images. This ensures that the potential for matching interest points exists, provided a feature matching scheme is capable of associating the interest points.

For the purposes of this thesis, the feature detectors used are those suggested by the authors of feature matching schemes. The pairing of feature detectors with matching algorithms to improve the feature matching performance is beyond the scope of this work.

### 2.2.2 Feature Matching

The key component in feature-based object recognition is feature matching. Here, features in a new scene are identified and compared with previously extracted features of an object. Should sufficient feature matches be found, the object is assumed to be present in the scene. Unfortunately, since detected features are nothing more than image coordinates, more information about features is required before matching can take place. This information is typically encoded in a descriptor[5], and matching occurs by finding highly similar descriptors.

The choice of descriptors and the information they contain vary greatly across applications. One of the simplest descriptors available merely involves the use of an image patch surrounding a feature. Unfortunately, while this image patch often contains all the information required for matching, further processing is generally required for it to be used effectively. Mikolajczyk and Schmid (2005) provide a comparison of some of the most popular feature descriptors available. The Scale Invariant Feature Transform (SIFT), proposed by Lowe (2004), is one such approach and generally considered the de facto standard of feature matching in computer vision, producing excellent matching results over extreme viewing conditions.

The SIFT matching scheme detects features by selecting the maxima and minima of the difference between images at successive scales. This difference-of-Gaussians keypoint selector, initially proposed by Lowe (1999), is more accurately classed as a blob detector and lacks the localisation properties of feature detectors, but is better suited to the SIFT feature matching scheme as the robust detection allows better matching over more extreme conditions. It is important to note that the operation of the SIFT detector is very similar to that of the Harris feature detector.

After keypoint detection, an orientation and gradient magnitude, based on pixel differences around the keypoint, is assigned to each feature. Image gradients are then computed at sub-regions around each keypoint and used to vote for a general region orientation. These orientations are then accumulated into a $4 \times 4$ array of histograms, each consisting of 8 orientation bins. This 128 bin feature vector makes up the SIFT descriptor. Orientations in the descriptor are measured relative to the keypoint orientation so as to ensure rotation invariance.

Feature matching takes place by performing a nearest neighbour search on descriptors. Unfortunately, the large feature vector makes SIFT extremely slow and ill suited to real-time processing. Beis and Lowe (1997) proposed an approximation to the nearest neighbour search called Best-Bin-First, which speeds up the algorithm over large databases, but is still too slow for real-time applications. Further attempts to improve the processing speed by reducing

[5]A descriptor is a collection of properties describing the area around a keypoint detected in an image.

the dimensionality of the feature vector through Principal Component Analysis have been made by Ke and Sukthankar (2004), who proposed the PCA-SIFT algorithm.

The low speed of the SIFT algorithm and its direct derivatives prompted Bay *et al.* (2008) to suggest the Speeded Up Robust Features (SURF) detector-descriptor scheme, a high speed approximation of the SIFT approach. SURF improves the speed of SIFT dramatically by approximating the difference-of-Gaussians operator with box filters and the SIFT gradient calculations with a Haar wavelet. These improvements in speed make SURF a candidate for use in a human-following system, given the real-time requirements of visual servo control. Properties of the SURF technique are discussed in much greater detail in Section 3.2, together with an in-depth treatment of its operation.

The detector-descriptor approach suffers extensively from the trade-off between speed and descriptor information content. Lepetit and Fua (2006) proposed an extremely fast alternative to computing a descriptor around each keypoint, formulating the matching problem as a classification problem. Their approach requires a training step, so is only applicable in cases where knowledge of the target object to be recognised is available beforehand. Initially a set of views of the target object is generated, by applying known noise and distortions. Keypoints are located in each view and those not present in all the views are discarded as they represent poor candidates. This ensures that only good keypoints are retained and allows for a simple feature detector such as FAST to be used, as the performance of the feature detector does not impact the matching process significantly.

The remaining keypoints are then classified using randomised trees with node tests based on the difference of intensities between two pixels near respective keypoints, an approach similar to that of Amit and Geman (1997). This randomised tree approach is extremely fast and produces good matching results. Poor matches could potentially occur, however, and an additional stage where incorrect matches are discarded is required for practical applications.

The randomised tree approach to feature matching was reformulated as a semi-naive Bayesian classification problem by Ozuysal *et al.* (2010). This reformulation resulted in a feature matching scheme that produced better results than randomised trees when a larger training set was utilised. This approach is well suited to the human-following problem as it produces good results extremely quickly, and the requirement for an initial training stage is of no concern in this task. More detail regarding the operation of this feature matching scheme is provided in Section 3.2.

Thus far, only a class of feature matching systems based on recognition have been considered. A second class of matching systems performing recognition through tracking needs mentioning. Here, the objective is to track the motion of features detected in a template image, eliminating the need for lengthy matching processes.

The most popular feature tracking algorithms rely on some form of optical flow to track features from frame to frame. Optical flow is the pattern of apparent motion in an image scene caused by the relative motion between a camera and the scene. A comparison of the various techniques for estimating optical flow provided by Barron *et al.* (1994) found that the Kanade-Lucas tracker (Lucas and Kanade (1981)) was the most reliable approach of those available at the time.

The features of Shi and Tomasi (1994) are specifically designed for use with the Kanade-Lucas tracker (KLT). The KLT divides images into a set of small windows and assumes a locally constant optical flow over these. This assumption results in an over-specified set of equations governing individual pixel optical flow. These equations are solved in a linear least squares method by calculating image derivatives along all dimensions.

The KLT is suited to a target-following application as it is extremely fast, but lacks the recognition performance of direct matching algorithms. More detail on the KLT operation is provided in Section 3.2.

## 2.3 Pose Estimation

Position-based visual servoing techniques require that the pose of a target object detected in a scene be determined prior to the application of a particular control algorithm. The process of determining the pose of a target object, given a set of feature correspondences in two image scenes, is referred to as pose estimation.

As many pose estimation algorithms rely on perspective camera models, a brief introduction to projective and perspective camera geometry is provided in Appendix A.1. Further information on the geometry of computer vision can be obtained from the excellent text of Hartley and Zisserman (2004).

Figure 2.2 depicts the general feature-based pose estimation problem. Given a set of corresponding features in two image scenes, the goal is to find the camera rotation and translation between viewpoints. This transformation then provides target object pose, measured relative to the initial viewpoint.

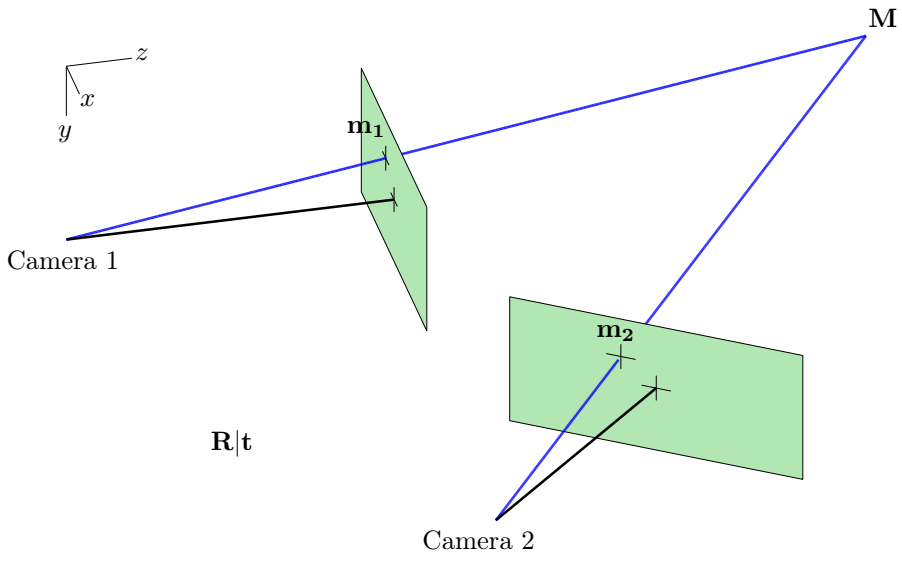



Figure 2.2: A feature at world coordinate **M** is projected onto two images, captured from differing viewpoints, at image plane coordinates $\mathbf{m_1}$ and $\mathbf{m_2}$ respectively. Using this information, or subsets thereof, the camera rotation **R** and translation **t** that moves the images into a common reference frame needs to be found.

### 2.3.1 3D - 3D Pose Estimation

The simplest form of pose estimation is one where the 3D locations of feature correspondences are known. Here, the pose estimation problem reduces to the problem of finding the rotation and translation that causes the 3D features of a template or reference object to best overlap with those of the recognised object in a different scene. This is a relatively easy task and can be performed through the application of Procrustes[6] analysis. The Procrustes orthogonal problem, solved by Schönemann (1966), is that of finding an orthogonal matrix mapping between two matrices in linear algebra.

Schönemann (1966) showed that if **x** and **y** are matrices consisting of the 3D co-ordinates of features in the respective scenes, they can be related by a scale $\lambda$, rotation **R** and translation **t** such that

$$\mathbf{x} = \lambda \mathbf{R}\mathbf{y} + \mathbf{t}. \tag{2.3.1}$$

[6]In Greek mythology, Procrustes was a bandit who invited passers by to stay in a bed in his stronghold, and then stretched them on the rack or amputated body parts so that they would fit (Plutarch (1914)). The similarities between this and the problem of transforming point clouds onto a common axis to ease their comparison led to the term statistical Procrustes analysis.

Appendix B.1 shows how the scale, rotation and translation matrices are calculated by applying singular value decomposition. This calculation is a single pass approach and does not provide the best solution in the presence of noise introduced by inadequate feature localisation. Zhang (1992) introduced the Iterative Closest Point (ICP) algorithm which improves on the Procrustes estimate by recursively estimating the transformation between feature point clouds.

Unfortunately, determining the 3D locations of features requires at least two cameras in a stereo configuration, or that the relative distance between features be known beforehand. In this work, the use of multiple cameras is both undesirable and unnecessary, while the requirement that the relative distance between features be known restricts the pose estimation problem to rigid objects.

### 2.3.2 2D - 3D Pose Estimation

The requirement that the 3D locations of features in both image scenes be known can be relaxed slightly, by considering a set of techniques that find the transformation between the image coordinates of the detected object and known 3D coordinates of the target in its reference configuration. These approaches are more complex than 3D - 3D pose estimation techniques as they need to take perspective camera effects into account while estimating the transformation.

The mapping between world and image plane can be described by camera intrinsic, rotation and translation matrices, as discussed in Appendix A.1:

$$\begin{bmatrix} x \\ y \\ \lambda \end{bmatrix} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.3.2}$$

Here, $[X, Y, Z]^{\mathrm{T}}$ are the 3D world coordinates projected onto the image plane at image coordinates $[x/\lambda, y/\lambda]^{\mathrm{T}}$. $\mathbf{K}$ is the intrinsic camera matrix obtained through a calibration process such as the one described in Appendix A.2. The rotation matrix $\mathbf{R}$ contains three degrees of freedom corresponding to the roll, pitch and yaw motions used to align the camera centred axis with a reference frame. The aligned frame is then shifted into place using the three degree of freedom translation matrix $\mathbf{t}$.

The 2D - 3D pose estimation techniques simply reduce to a problem of solving for the six degrees of freedom in the rotation and translation matrices, given the world and image plane coordinates. It is clear that this is possible if at least six non-coplanar feature correspondences are available, but solutions using $n$ points have been presented, leading to the 2D - 3D pose estimation problem being termed the perspective-$n$-point or P$n$P problem by Fischler

and Bolles (1987). Quan and Lan (1999) presented a family of linear $n$-point solutions, emphasising the conditions under which a unique solution to the problem can be obtained.

In general, far more than six features are available, and the problem is solved in a least squares sense using many points to counter noise, so the benefit of solutions using a minimal number of points is unclear. Often, however, not all feature correspondences are good matches and a robust solution is required. Many robust approaches, such as the Random Sample Consensus (RANSAC) algorithm proposed by Fischler and Bolles (1987), attempt to find the best set of matches with which to solve the problem, and iterate over different subsets of feature matches in doing so. Reducing the required number of feature matches for each potential solution can vastly improve on the algorithm run time in such an iterative approach.

DeMenthon and Davis (1995) introduced a method, termed Pose from Orthography and Scaling with Iterations (POSIT), which determines the pose of an object using four or more non-coplanar points. Object pose is initially estimated by approximation of the perspective camera projection with a scaled orthographic projection and then improved through an iterative process. The POSIT algorithm is notable as it does not require an intrinsic camera matrix. Extensions to the POSIT algorithm that allow the pose of an object to be determined when the correspondences between features are not known have been made by David *et al.* (2002).

Object pose can also be determined without knowledge of an intrinsic camera matrix by performing camera calibration each time a target object is detected in an image scene. This calibration process is discussed in great detail by Hartley and Zisserman (2004). Initially, the camera projection matrix is estimated using six or more point correspondences. Thereafter, the camera calibration, rotation and translation matrices are extracted through QR decomposition.

While notable, the ability to determine pose without requiring an intrinsic camera matrix is of no particular importance in a target-following application where knowledge of the target and camera can be made available beforehand. Although possible, obtaining knowledge of the 3D locations of features on a target object prior to the following task is not necessarily desirable though, and a pose estimation scheme that operates directly on the 2D locations of feature points in the target and template image is preferred.

### 2.3.3 2D - 2D Pose Estimation

2D - 2D pose estimation with a moving camera forms part of a collection of techniques known as structure from motion. Here, a mapping between correspondences in views is used to extract the scene structure and camera motion. Although this definition appears to refer to the act of building up the structure of a static environment using a moving camera, it also applies to the

task of determining the structure and relative motion between an object and camera.

The primary objective in structure from motion is to solve for the essential matrix $\mathbf{E}$ which satisfies the condition $\hat{\mathbf{x}}_1^{\mathrm{T}}\,\mathbf{E}\,\hat{\mathbf{x}}_2 = 0$ for any pair of corresponding normalised points $\hat{\mathbf{x}}_1$ and $\hat{\mathbf{x}}_2$ in two image scenes (Hartley and Zisserman (2004)). A normalised point is one which has had the intrinsic camera effects removed through multiplication by the inverse of the camera calibration matrix,

$$\hat{\mathbf{x}} = \mathbf{K}^{-1}\mathbf{x}. \tag{2.3.3}$$

The most common approach to estimating the essential matrix solves the problem in a least squares manner, using at least eight point correspondences. The eight-point algorithm is discussed in detail by Hartley and Zisserman (2004). As in the case of 2D - 3D pose estimation, typically more than eight points are used and the problem solved robustly. Less point correspondences can be used to solve for the essential matrix by incorporating motion constraints, with the benefit of faster algorithms.

Despite having six unknowns (three rotation angles and three translation values), the essential matrix has five degrees of freedom. This is due to a scale ambiguity, caused by the homogeneous representation of image coordinates. Hartley and Zisserman (2004) show how the essential matrix can be factorised to obtain four camera projection matrix solutions, from which rotation and translation components are easily extracted. Finally, a single solution is selected by taking into account a logical constraint in computer vision: the object viewed must be in front of the image planes.

The structure from motion problem can be simplified by incorporating a prior of expected structure. Faugeras and Lustman (1988) show how the use of a piecewise planar approximation to the environment eases the calculation of camera motion. The assumption that objects are planar may seem limiting, but this is not always the case, as explained in Section 3.4. If $\mathbf{x}_1$ and $\mathbf{x}_2$ are the homogenised coordinates of corresponding features from two image scenes, they are related by a homography $\mathbf{H}$ as

$$\mathbf{x}_1 = \mathbf{H}\,\mathbf{x}_2. \tag{2.3.4}$$

Hartley and Zisserman (2004) show that coplanar points in two images are all related by the same homography and show how this homography can be estimated using the direct linear transform (DLT). The work of Faugeras and Lustman (1988) is notable as it provides a means of decomposing a homography and extracting the physical 3D rotation and translation between planes related by the homography. A similar decomposition is presented by Ma *et al.* (2003). The homography-based approach to pose estimation is discussed in detail in Section 3.3 as it is directly applicable to the target following problem. The DLT algorithm and Faugeras decomposition are listed in Appendices C.1 and C.2 respectively.

## 2.4 Target Tracking

Typically, pose estimates are noisy and require additional refinement prior to use as set-points in a control algorithm. Refinement usually takes place through the application of a tracking algorithm or filter with a predictive component. The Kalman filter (Kalman (1960)) is a useful tool for fusing state measurements with predicted behaviour and is commonly used in autonomous navigation systems. Thrun *et al.* (2005) provide an excellent discussion on Kalman filtering in the context of mobile robotics.

The extended Kalman filter (EKF), McGee and Schmidt (1985), is a non-linear extension to the Kalman filter, which allows for state estimation by linearising models about the current estimate. More details regarding the extended Kalman filter can be found in Appendix D.1.

The extended Kalman filter operates under the assumption that measurement and prediction noise is Gaussian. In some cases, this assumption is invalid and a better approach is to represent noise distributions using particles. The operation of particle filters is easily understood by considering a particular variant, the sequential-importance-sampling re-sampling (SIS-R) particle filter algorithm, discussed in detail by Ristic *et al.* (2004).

At first, particles are initialised randomly according to prior knowledge or from a uniform set. At each time step, a motion model predicts the state of each particle. An observation model then assigns a weight to each particle and a best estimate of the current state is drawn from the particle distribution. Finally, a new set of equally weighted particles is selected by re-sampling the particle distribution.

While the particle filter is able to approximate non-Gaussian noise distributions better than the extended Kalman filter, it is significantly slower as motion and observation models need to operate on multiple particles. Moreover, the memory requirements of particle filter algorithms are excessive, as the performance of the algorithm is dependent on the number of particles used. The computational complexity of particle filter algorithms has caused a preference of the extended Kalman filter for real-time applications, and is widely acknowledged as the workhorse of navigation (Levy (1997)).

## 2.5 Motion Control of Wheeled Robots

Problems relating to the motion control of wheeled robots have been studied extensively and as a result a well established set of control strategies are in use. A review of the most common and more effective strategies for the motion control of non-holonomic[7] platforms is presented by Morin and Samson (2008).

[7] A non-holonomic vehicle is a vehicle with less controllable degrees of freedom than the total available degrees of freedom.

Wheeled robotic platforms are typically modelled as either unicycle-type robots or car-like robots. Unicycle-type robots have two drive wheels, on the same axis, which can be driven independently, while car-like robots are typically rear wheel drive with a front wheel steering assembly. Both kinematic and dynamic control models are used in conjunction with these chassis models, but since many platforms already include basic dynamic control, there is a tendency to make use of higher level kinematic control models and to decouple kinematics and dynamics.

The kinematic models of unicycle-type and car-like robots are discussed in the review of Morin and Samson (2008) and shown in Equations 2.5.1 and 2.5.2 respectively, using Newton's notation:

$$\dot{\mathbf{X}}_{\mathbf{uni}} = \begin{cases} \dot{x} = u_1 \cos(\theta) \\ \dot{y} = u_1 \sin(\theta) \\ \dot{\theta} = u_2 \end{cases} \tag{2.5.1}$$

$$\dot{\mathbf{X}}_{\mathbf{car}} = \begin{cases} \dot{x} = u_1 \cos(\theta) \\ \dot{y} = u_1 \sin(\theta) \\ \dot{\theta} = \frac{u_1}{l \tan(\phi)} \\ \dot{\phi} = u_2 \end{cases} \tag{2.5.2}$$

Here, $x$ and $y$ represent the position of the centroid on the drive wheel axis, $\theta$ the orientation of the robot, $l$ the distance between front and rear axles in a car like robot model, and $\phi$ the orientation of the steering mechanism in a car like model. These parameters are illustrated graphically in Figure 2.3.

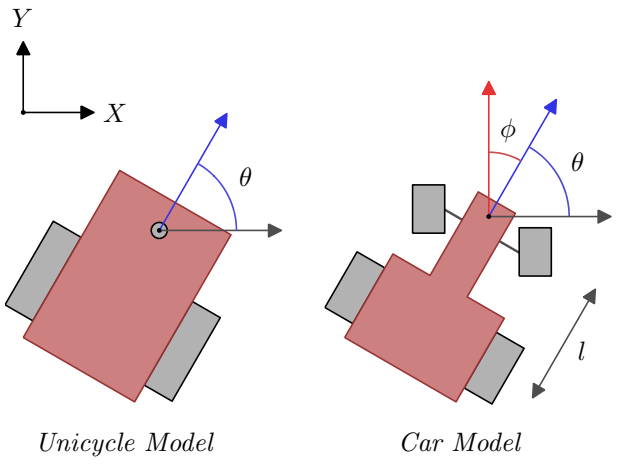



Figure 2.3: A graphical interpretation of the variables $x$, $y$, $\theta$, $\phi$ and $l$, which parametrise the two motion models commonly used to describe wheeled platforms.

The control systems implemented in this thesis all assume a unicycle model, as the target following is implemented on a differential, skid steer[8] platform. They should, however, directly extend to the control of car-like platforms. The control inputs $u_1$ and $u_2$ in the unicycle model are the skid steering platform's forward and angular velocity, and hereafter are referred to as $v$ and $\omega$ respectively, for greater clarity.

The control of wheeled robots described by these models has been divided into two strategies: simple point-to-point positioning without control of orientation and direction-based motion control, taking orientation into account.

Direction-based motion control allows for the tracking of a reference trajectory provided it is feasible. A trajectory is considered feasible if it solves the robot's kinematic model for a set of control inputs, i.e. the trajectory to be followed could have been created by a vehicle with the same kinematics as the robot to be controlled (Morin and Samson (2008)). Various control strategies that asymptotically stabilise feasible reference trajectories are discussed in the work of Morin and Samson (2008), Petrov and Parent (2006) and Ma *et al.* (1999).

The work of Petrov and Parent (2006) is of particular interest as the authors apply an adaptive control strategy to the problem of reversing a platoon of vehicles where the lead vehicle undergoes constant velocity manoeuvres. This is accomplished by using a motion model incorporating bearing measurements of the lead vehicle, which are extracted from visual data or a suitable range sensor. In contrast, Ma *et al.* (1999) perform control in the image directly, by extending the kinematic model of the vehicle to one which includes the dynamics of ground plane curves.

Point-to-point positioning, which does not take relative orientation into account, is the most common approach for moving towards set-points on non-feasible trajectories. Here, platforms merely move directly towards set-points, with orientation uncontrolled.

Maya-Mendez *et al.* (2006) propose a direction-based control strategy which forgoes asymptotic stability and instead moves towards practical stabilisation, a strategy which allows for small tracking errors so that non-feasible reference trajectories can be tracked. This approach essentially causes a platform to move towards various set points sampled from the trajectory, along paths which potentially deviate from the original non-feasible trajectory, but in such a way as to cause the end pose of the platform to be near that originally extracted from the trajectory.

[8]Skid steering refers to the turning of a vehicle by adjusting the speed of wheels or tracks on opposite sides of a platform. Note that this definition differs slightly from that of a differential drive vehicle, since skid-steered vehicles have multiple wheels or tracks on each side of the platform.

## 2.6 Related Work

Thus far, emphasis has been on describing the individual components which can be used to build up a human-following system. Complete approaches, solving the human-following problem in its entirety, are now considered, along with the challenges human following entails.

Almost all vision-based human-following systems use a position-based visual servo control approach, with the human initially detected and located in a 3D coordinate frame. Platform navigation is considered a separate task, with the robot controlled to move towards the measured human position.

Early examples of vision-based human-following robots made use of simple template matching schemes (Hirai and Mizoguchi (2003)) or colour-based blobs with contour models (Schlegel *et al.* (1998)) for target detection and recognition. The latter approach uses stereo-vision in order to obtain an absolute scale representation of the human's position. In the single camera case, however, absolute scale is typically not available and alternative distance measurements, such as the size of detected blobs (Latif *et al.* (2009)), are incorporated for navigation.

These approaches, and other early ones, suffer potential target ambiguity in the presence of multiple persons or cluttered environments, mainly due to detection and recognition schemes that are not particularly robust. More recently, feature-based approaches have been applied to the problem with impressive results presented by Chen and Birchfield (2007).

Purely vision-based human-following systems typically lack the following performance of those using multiple sensors and fused layers of detectors to maximise available target information. Examples of human followers using multiple sensors include the work of Germa *et al.* (2009), who combined RFID trackers with complementary visual refinement, and Dai *et al.* (2008), who boosted their visual tracking with laser leg detection. For the purposes of this thesis, however, only the vision component of multi-layered trackers is considered, with the option to expand left as future work.

The above-mentioned approaches, and many others, merely use some form of position measurement for navigation. More intelligent navigation schemes could be implemented, however, if some knowledge of the intended motion of the human target was incorporated. In fact, results of a preliminary study on the social acceptance of human-following approaches (Gockley *et al.* (2007)) indicate that the following of direction is more acceptable to people than point-to-point path following.

The work of Gockley *et al.* (2007) used LIDAR to detect a human target, building human trajectories by analysis of human motion over time. The platform was then controlled to move along the same trajectories, assuming they were feasible. This strategy was compared to one in which the platform moved directly to targets, through an interview process with followed humans.

While building up a trajectory using only positional information is effective,

a potentially improved trajectory could be generated if human orientation information is included in some way. A human-following system that includes orientation information requires that some measure of human body pose be made.

Unsurprisingly, human pose estimation is a popular topic within the computer vision research community and a large body of work on the subject is available. While many pose estimation algorithms rely on multiple images of static scenes with moving humans for pose estimation, a human following robot requires a pose estimation system that is able to extract target pose in images captured from a moving camera.

Common approaches to human pose estimation, such as that of Chen *et al.* (2009), fit complex articulated body models to image scenes. Lee and Cohen (2004) combine articulated models with shape and clothing models to detect upper body pose in single images. Their approach is iterative, with a hypothesis and test framework used to fit models to image scenes.

Agarwal and Triggs (2004), however, reject the model-based approach to human pose estimation, in favour of an approach that directly fits joint angles to silhouette shape descriptors using nonlinear regression. This approach is effective, but only provides body joint angles, with the orientation of body parts not explicitly extracted. Recently, Ferrari *et al.* (2009) proposed a human pose estimation scheme for use on unconstrained image sequences in television. Their approach aims to extract the position and angle of each body part detected in an image. This is accomplished by progressively reducing the search space in images. Initially, a simple detection process finds the torso in the image, after which a more advanced model searches for the location of limbs and other body parts. Note that this approach finds the 2D locations and angles of body parts and provides no 3D orientation information.

While these and similar techniques are effective and produce commendable results, the focus in human-following tasks is on simplicity and speed. Moreover, these techniques typically produce a large amount of information, such as individual limb positions, where a single three-dimensional orientation angle would prove sufficient for the purpose of controlling a wheeled robot.

A broad overview of the components and approaches to robotic human followers has been presented here. In Chapter 3, theoretical analysis of the specific approaches implemented and design of the proposed solution are presented.

# Chapter 3

# Theoretical Design and Analysis

The theoretical design and the analysis of the components and systems used by the human-following robot are discussed here. Initially, the design of a generalised planar target-following robot is presented, followed by an extension to the human-following case.

## 3.1 Visual Servo Control Strategy

A decision needs to be made as to the underlying visual servo control strategy used, before the complete system design can be presented. Theoretical analysis is initiated by the evaluation of IBVS control techniques through simulation. This evaluation is used to determine the suitability of IBVS control systems to the target-following problem, given the benefits of PBVS methods, and to aid in the choice of visual servo control strategy.

### 3.1.1 IBVS Simulations

The simulation presented here takes place in 3D space, under the assumption of a six degree of freedom (6 DOF) camera. This is far less constrained than the two degrees of freedom associated with a wheeled terrestrial vehicle, but serves to illustrate the properties of IBVS control systems when platform dynamics are not included.

Chaumette and Hutchinson (2008) show that if $\mathbf{x} = (x, y)$ represents the image coordinates of a projected feature at world coordinates $(X, Y, Z)$, the feature motion induced by camera motion is given by the product of the camera velocity screw $\mathbf{v_c}$ and a matrix $\mathbf{J_x}$, termed the interaction matrix:

$$\dot{\mathbf{x}} = \mathbf{J_x v_c}. \tag{3.1.1}$$

The velocity screw $\mathbf{v_c} = \begin{bmatrix} \dot{t}_x, & \dot{t}_y, & \dot{t}_z, & \omega_x, & \omega_y, & \omega_z \end{bmatrix}^{\mathrm{T}}$ represents the possible motion of the 6 DOF camera, three translational velocities along Cartesian axes

and three rotational velocities about the axes. These axes represent a right-handed coordinate system centred on the camera image plane, with $t_z$ pointing along the optical axis. The velocity of features, $\dot{\mathbf{x}} = [\dot{x},\ \dot{y}]^{\mathrm{T}}$, is represented in Newton's notation.

The interaction matrix is derived in the work of Chaumette and Hutchinson (2008) and shown to be

$$\mathbf{J_x} = \begin{bmatrix} \frac{f}{Z} & 0 & -\frac{x}{Z} & -\frac{xy}{f} & \frac{f^2+x^2}{f} & -y \\ 0 & \frac{f}{Z} & -\frac{y}{Z} & \frac{f^2+y^2}{f} & \frac{xy}{f} & x \end{bmatrix}, \tag{3.1.2}$$

with the assumption that the focal length $f$ is equal along both the $x$ and $y$ axes. The Jacobian of Equation 3.1.2 only relates camera motion to the velocity of a single feature. In practice, multiple features are present and interaction matrices are stacked into a single, large Jacobian $\mathbf{J}$, associated with a set of $n$ stacked feature velocities:

$$\begin{bmatrix} \dot{x_1} \\ \dot{y_1} \\ \dot{x_2} \\ \dot{y_2} \\ \vdots \\ \dot{x_n} \\ \dot{y_n} \end{bmatrix} = \begin{bmatrix} \frac{f}{Z_1} & 0 & -\frac{x_1}{Z_1} & -\frac{x_1y_1}{f} & \frac{f^2+x_1^2}{f} & -y_1 \\ 0 & \frac{f}{Z_1} & -\frac{y_1}{Z_1} & \frac{f^2+y_1^2}{f} & \frac{x_1y_1}{f} & x_1 \\ \frac{f}{Z_2} & 0 & -\frac{x_2}{Z_2} & -\frac{x_2y_2}{f} & \frac{f^2+x_2^2}{f} & -y_2 \\ 0 & \frac{f}{Z_2} & -\frac{y_2}{Z_2} & \frac{f^2+y_2^2}{f} & \frac{x_2y_2}{f} & x_2 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \frac{f}{Z_n} & 0 & -\frac{x_n}{Z_n} & -\frac{x_ny_n}{f} & \frac{f^2+x_n^2}{f} & -y_n \\ 0 & \frac{f}{Z_n} & -\frac{y_n}{Z_n} & \frac{f^2+y_n^2}{f} & \frac{x_ny_n}{f} & x_n \end{bmatrix} \begin{bmatrix} \dot{t}_x \\ \dot{t}_y \\ \dot{t}_z \\ \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \tag{3.1.3}$$

The knowledge of the effects of camera motion on image feature motion allows the design of an appropriate control law. Ideally, the controller should undo all the perspective camera effects and cause the camera to move so as to position features in a desired configuration. This is accomplished by inverting the Jacobian, $\mathbf{J}$, which leads to the feedback control law

$$\mathbf{v_c} = \mathbf{J}^{+}\dot{\mathbf{x}}^{p}, \tag{3.1.4}$$

where

$$\dot{\mathbf{x}}^{p} = -\gamma\left(\mathbf{x_s} - \mathbf{x_s}^{d}\right). \tag{3.1.5}$$

Here, $\mathbf{x_s}$ is the set of measured stacked feature positions, and $\mathbf{x_s}^d$ the set of desired stacked feature positions. $\gamma$ is a proportional control gain used to weight the amount of control action used. The vision system is a first order process, so the gain relates directly to the system time constant, and is selected so as to obtain a desired response speed. $\mathbf{J}^{+}$ is the Moore-Penrose pseudo-inverse of the Jacobian $\mathbf{J}$.

The operation of the control law is now shown through the use of a Simulink simulation. The camera focal length is assumed to be unity for convenience,

and only four hypothetical image features are used. Chaumette and Hutchinson (2008) show that at least three points are required to control the six degrees of camera freedom, but that more than three points are needed to correctly distinguish between camera poses which may have caused a particular feature configuration.

Figure 3.1 shows the results of a hypothetical feature positioning task where features are controlled to move from a fronto-parallel position to an arbitrary set of desired positions. Feature positions are localised perfectly and it is assumed that there are no transport delays in the system. A relatively large gain of 50 was used in these simulations, to show that a fast response time is achievable in the ideal case.

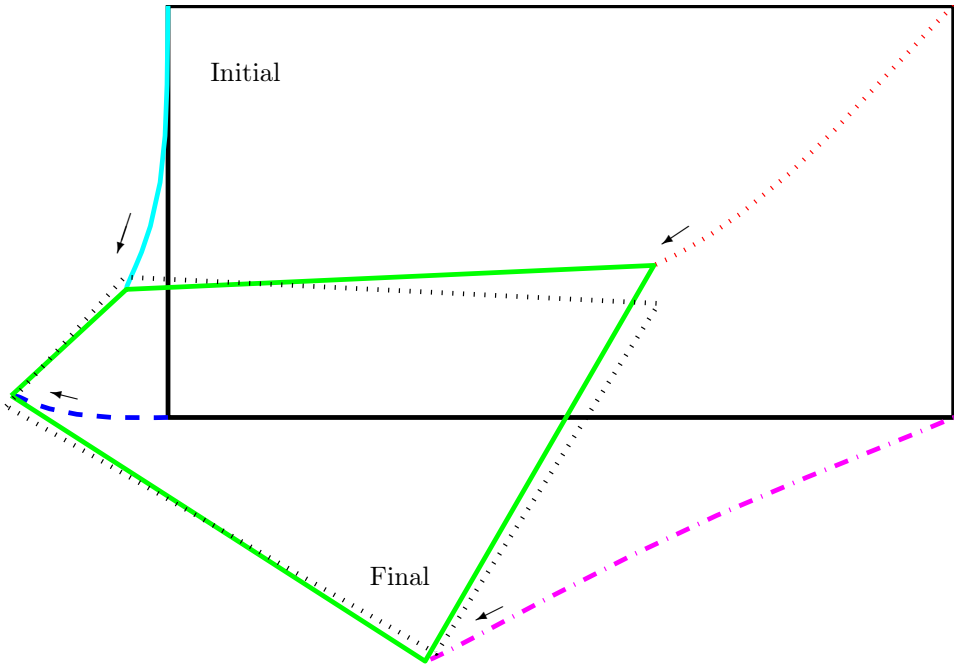



Figure 3.1: The motion of features for ideal IBVS control is shown for a hypothetical control task. Each feature is represented by the corner of a coloured quadrilateral. The lines connecting corners show the paths followed by features. The desired end position of the features is marked by the dotted black quadrilateral.

The motion of features is quite smooth, a factor also indicated by Figure 3.2, which shows the difference between the desired and actual feature coordinates for each of the corners of the quadrilateral in Figure 3.1. Note that steady-state error is exhibited as the feature coordinate errors do not converge to zero. This is due to the fact that the control law tries to minimise all feature errors simultaneously, and the existence of non-zero feature error vectors in the null space of the Jacobian $\mathbf{J}$. This behaviour is clearly undesirable in a control system.

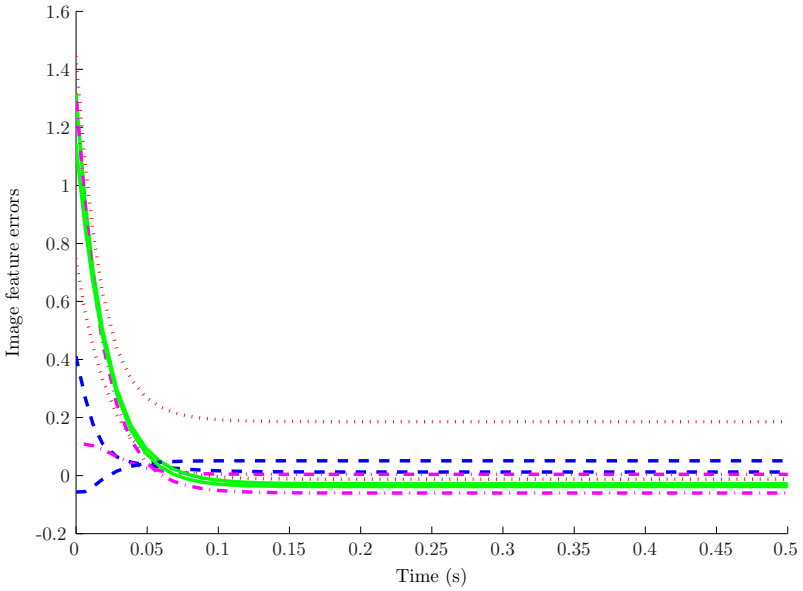



Figure 3.2: The positional errors of feature coordinates (both $x$ and $y$) as they converge to steady-state for a hypothetical control task when ideal IBVS control is used.

Figure 3.3 shows the real-world motion of the camera during the control simulation. The camera coordinates, together with the interaction matrices, are used by the simulator to calculate the new position of features after each control input.

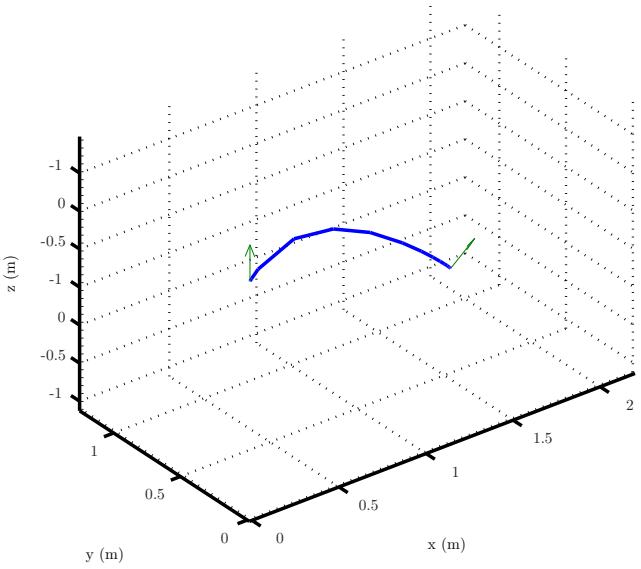



Figure 3.3: The path followed by the camera when ideal IBVS control is used. The green vectors show the orientation of the camera at the start and ending positions.

Obtaining the 3D coordinates of the camera, given a set of camera velocity inputs and a known starting position, is a rather complex process as the velocities relate to the camera body frame and not an external reference. As a result, the position and orientation of the camera needs to be obtained by applying various principles of inertial strapdown navigation, as described in Appendix E.1.

The simulations presented thus far have assumed that there are no transport delays in the system. In practice, the detection and matching of features is time consuming and this assumption invalid. Figure 3.4 shows the results of the same feature positioning task, but including a sampling delay of 80 ms, corresponding to an image processing frame rate of 12.5 fps, and designing the controller in the digital domain.

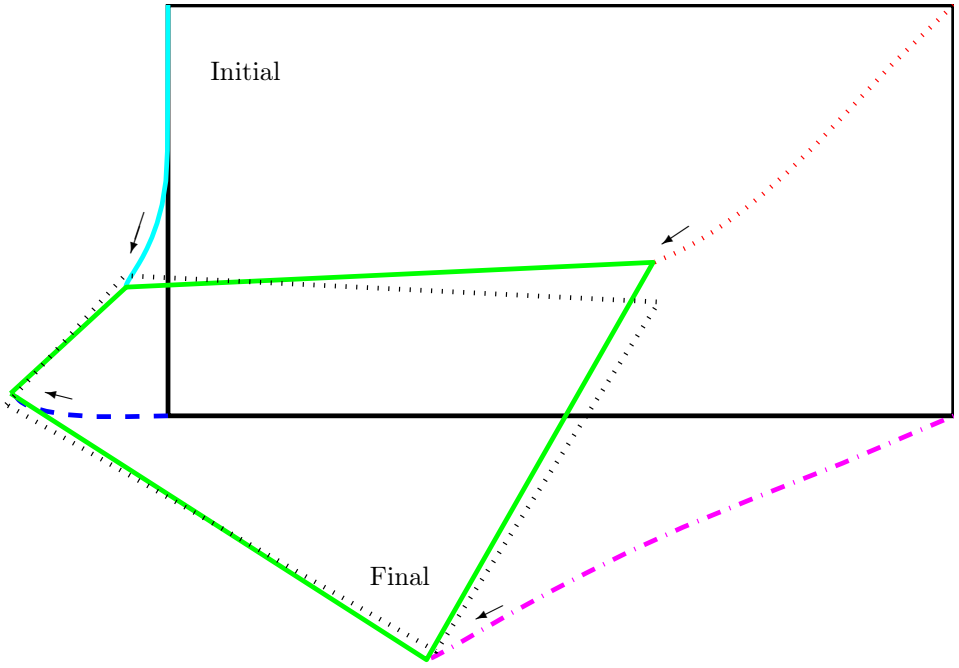



Figure 3.4: The motion of features for IBVS control with an 80 ms transport delay is shown for a hypothetical control task. Each feature is represented by the corner of a coloured quadrilateral. The lines connecting corners show the paths followed by features. The desired end position of the features is marked by the dotted black quadrilateral.

The motion of features is still smooth, and Figure 3.5 shows that the features do converge to a steady-state in a manner similar to that observed with no sampling delay.

Note that the features take significantly longer to converge to steady-state. This is due to the need to drop the proportional gain in the control law to 2.5 so as to reduce response speed enough to ensure stability, given the phase lag constraints of the added transport delays.

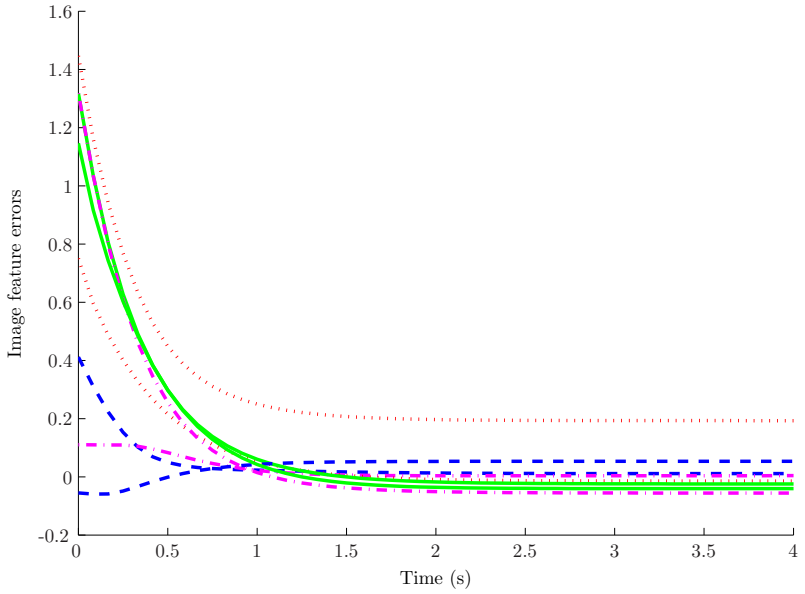



Figure 3.5: The positional errors of feature coordinates as they converge to steady-state for a hypothetical control task when IBVS control is used with an 80 ms transport delay.

Figure 3.6 shows the real-world motion of the camera during the control simulation with transport delays.

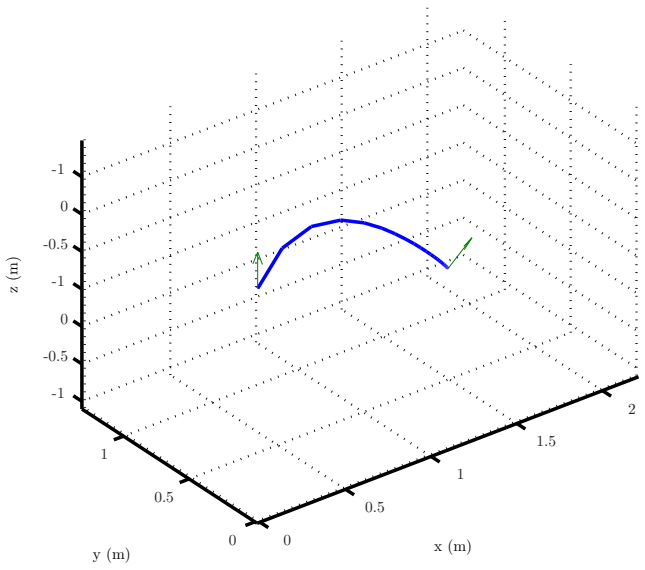



Figure 3.6: The path followed by the camera when IBVS control is used with an 80 ms transport delay. The green vectors show the orientation of the camera at the start and ending positions.

Analysis of the IBVS control law shows that it requires the distance of each feature from the camera, along the optical axis. This feature depth is not easily

obtained, as it requires at least two cameras in a stereo configuration or prior knowledge of the relative positions of image features on a rigid target in world coordinates, which is not readily available in target-following applications.

Alternatively, it may be possible to estimate the depth of features by including known camera motion constraints and solving pose estimation problems. Unfortunately, techniques such as these are time consuming, which has led to the use of a constant value for the depth of features in the control law, in an effort to avoid the process of depth approximation. Figure 3.7 shows the results of the feature positioning task, still including the transport delay of 80 ms, but using a constant depth approximation.

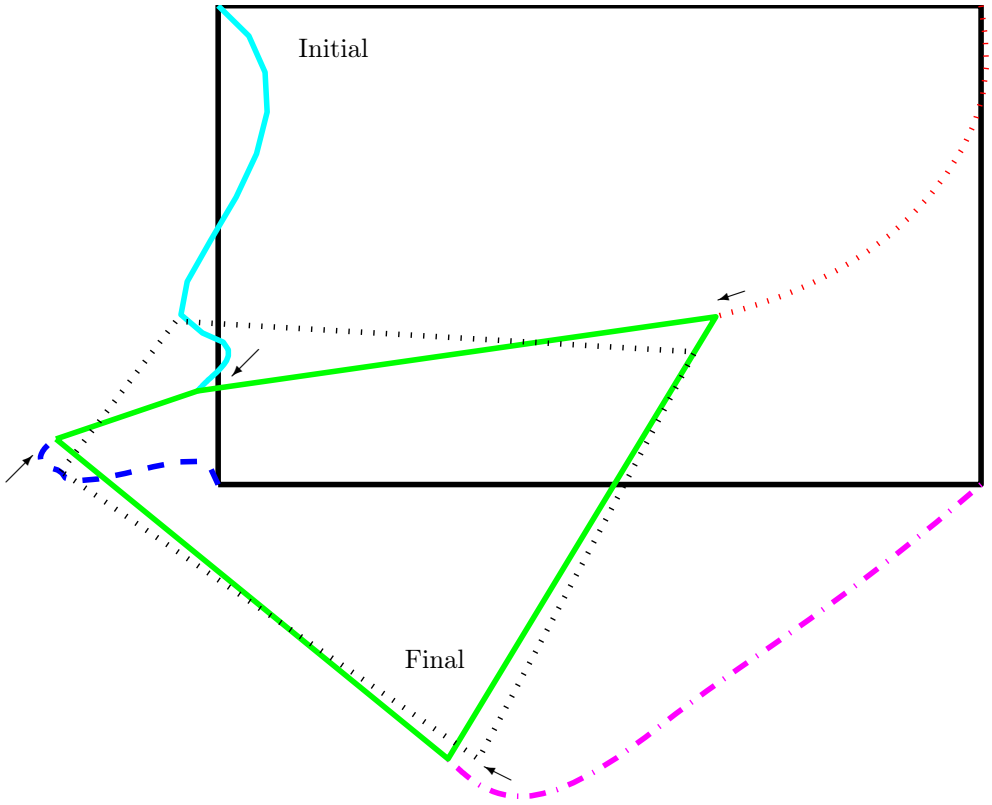



Figure 3.7: The motion of features for IBVS control with an 80 ms transport delay and a constant depth assumption is shown for a hypothetical control task. Each feature is represented by the corner of a coloured quadrilateral. The lines connecting corners show the paths followed by features.

The motion of features is no longer smooth, and Figure 3.8 shows that the steady-state errors have worsened.

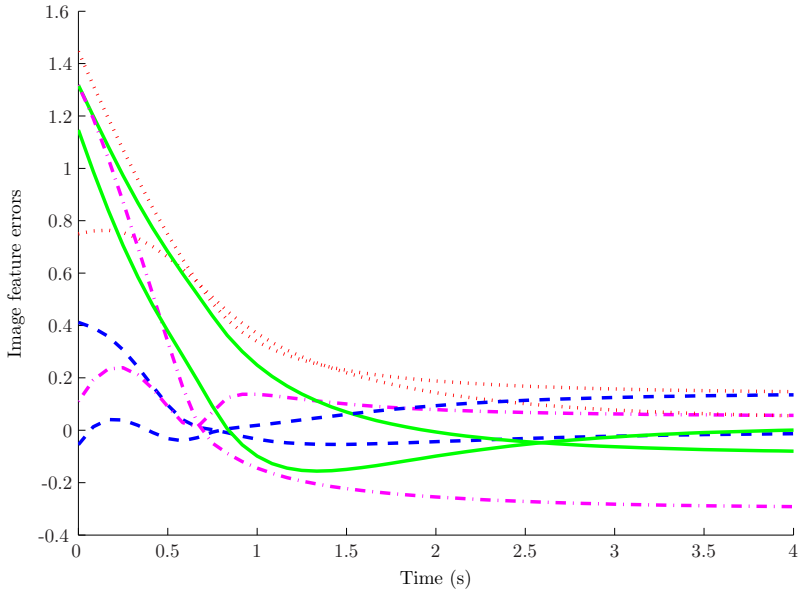



Figure 3.8: The positional errors of feature coordinates as they converge to steady-state for a hypothetical control task when IBVS control is used with an 80 ms transport delay and a constant depth assumption.

Figure 3.9 shows that the real-world motion of the camera during the control simulation.

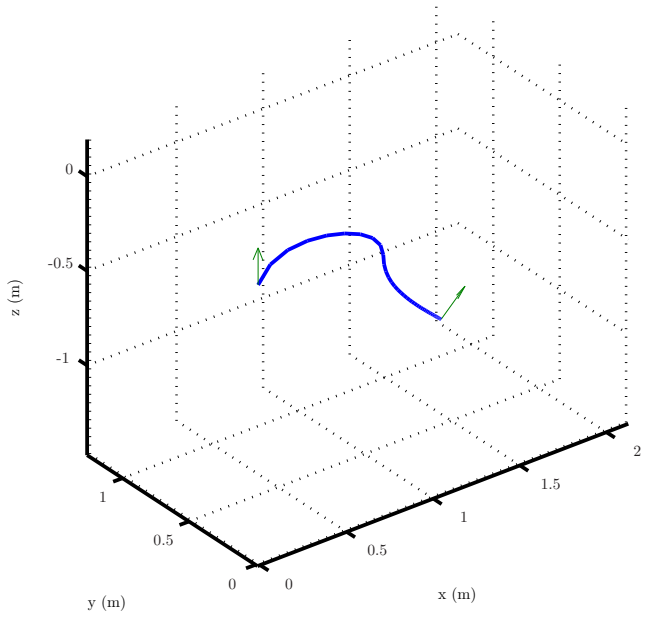



Figure 3.9: The path followed by the camera when IBVS control is used with an 80 ms transport delay and a constant depth assumption. The green vectors show the orientation of the camera at the start and ending positions.

The camera motion is no longer smooth, a factor which could potentially result in the loss of features due to motion blur.

Note that these simulations have assumed a stationary target, but are easily extended to incorporate moving targets without affecting the simulation results. Moreover, the effects of noisy or potentially incorrect feature locations has not been considered, but would worsen the control responses significantly.

### 3.1.2 Comparison of Visual Servo Control Strategies

The IBVS control simulations of Section 3.1.1 showed some of the limitations of a traditional IBVS control scheme. Undesirable camera motions can occur in practical systems because IBVS control only considers the motion of features in the image plane. Unlike PBVS control strategies, properly implemented IBVS control laws should ensure that features on a stationary target never leave the camera field of view, but in practice the jerky motion of the camera could cause the loss of the features from motion blur.

A primary argument for the use of an IBVS control system over a PBVS strategy is that there is no need for a pose estimation step. However, IBVS control laws require information about the depth of features, quantities that can only be obtained through a process similar to pose estimation. Although depth can be approximated by a rough estimate of the desired feature depth, this decreases the performance of the control algorithms, leading to increased response time and overall greater steady-state errors.

One challenge associated with PBVS control is that of stability[1]. As PBVS control is based on a pose measurement obtained through a highly non-linear, statistical pose estimation process, the stability of these control systems cannot be guaranteed. However, by ensuring that the relative pose between the target and camera remains in the most reliable and accurate pose estimation region through platform motion control, potential stability issues can be mitigated.

Though IBVS control successfully restricts control to the image plane, it does not easily extend to the inclusion of other sensors and navigation strategies. In contrast, the pose estimation stage of PBVS control produces target information which can be used for purposes other than control and also allows for the extension to aspects such as collision avoidance in a more advanced navigation scheme.

Due to these limitations of IBVS control, a PBVS control approach is used for the target-following problem. PBVS methods have the benefit of decoupling image processing and navigation tasks, which allows for smoother camera motion when used with a suitable control law. While the PBVS method

[1]In this context, the term stability differs slightly from that traditionally associated with control system design. Throughout this thesis, unless otherwise stated, a reference to stability refers to the ability of the control system to keep detected features bound within a camera's field of view.

does not ensure that features remain in view, this can be remedied through intelligent platform and camera motion control.

## 3.2 Recognition

Having selected a position-based visual servo control approach, the analysis now shifts to the individual components it requires. The first of these is feature-based recognition.

Three potential feature recognition approaches are discussed, and their matching results compared, given the requirements of a target-following application. In Section 2.2.2, the three feature recognition techniques, Speeded Up Robust Features (SURF), a semi-naive Bayesian classifier and Kanade-Lucas feature tracking were identified as suitable approaches to recognition in this context, due to their reported speed.

### 3.2.1 SURF Feature Matching

The operation of the SURF matching scheme, presented by Bay *et al.* (2008), is discussed first. The SURF matching scheme relies heavily on integral images, so an explanation of this concept is provided before explicit SURF matching details are presented.

An integral image allows for fast calculation of the convolution over rectangular regions. $I_{\Sigma}(x,\ y)$, the value of an integral image at coordinates $(x,\ y)$, is calculated as the sum of all pixels in a rectangle formed by the origin and the coordinates $(x,\ y)$ of the input image, $I$, or

$$I_{\Sigma}(x,\ y) = \sum_{i=0}^{x}\sum_{j=0}^{y} I(i,\ j). \tag{3.2.1}$$

The integral image only needs to be calculated once, but allows the sum of intensities in any rectangular region of an image to be calculated using only four additions (Figure 3.10). This means that the responses of box type filters can be calculated extremely quickly, as this is essentially a convolution operation.

The SURF feature detector is an approximation of the determinant of a Hessian matrix,

$$\mathcal{H}(x,\ y;\ \sigma) = \begin{bmatrix} L_{xx}(x,\ y;\ \sigma) & L_{xy}(x,\ y;\ \sigma) \\ L_{xy}(x,\ y;\ \sigma) & L_{yy}(y,\ y;\ \sigma) \end{bmatrix}. \tag{3.2.2}$$

Here, $L_{xx}(x,\ y;\ \sigma)$ is the result of the convolution between the image and the Gaussian kernel second-order derivative, $\frac{\partial^2}{\partial x^2} g(\sigma)$. Both $L_{xy}(x,\ y;\ \sigma)$ and $L_{yy}(x,\ y;\ \sigma)$ are calculated similarly. $\sigma$ is a scale parameter relating to the scale at which the operation is performed (see Appendix A.3 for a discussion

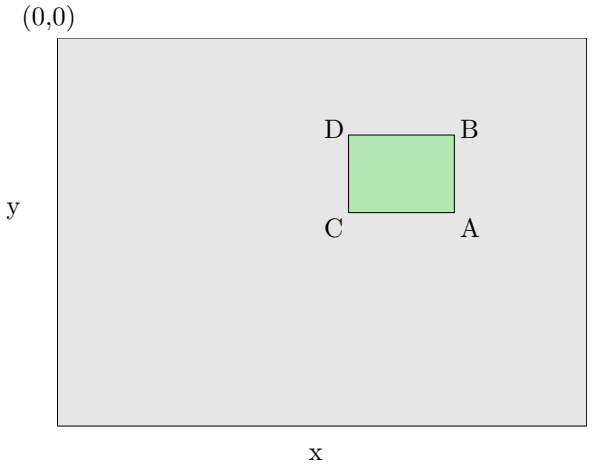



Figure 3.10: After an integral image has been calculated, the sum of intensities in a rectangular region is easily calculated using only four additions, $\sum = A - B - C + D$.

on scale space). Bay *et al.* (2008) use box filter approximations to these kernel derivatives to improve computational costs. Figure 3.11 shows the Gaussian second-order derivative kernels and their box filter approximations. Convolution of these box filters with an image results in the Hessian approximations, $D_{xx}$, $D_{xy}$ and $D_{yy}$.

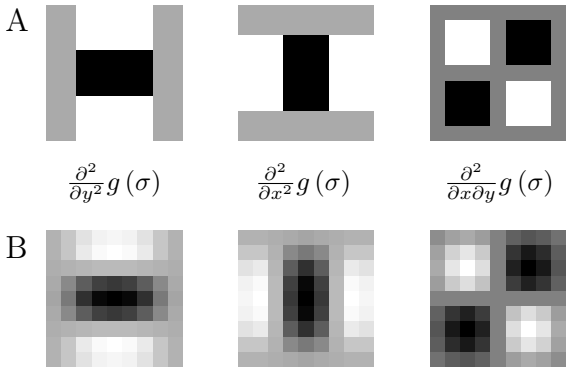



Figure 3.11: The Gaussian kernel functions (B) traditionally used for feature detection and the box filter approximations (A) of these utilised by the SURF algorithm.

The SURF detector is then written as

$$\det\left(\mathcal{H}_{\text{approx}}\right) = D_{xx}D_{yy} - \left(wD_{xy}\right)^2. \tag{3.2.3}$$

$w$ is a weight used to balance the determinant and shown by Bay *et al.* (2008) to be approximately 0.9 for $9 \times 9$ box filters. Filter responses are also normalised. As discussed earlier, in Section 2.2.1, scale invariance of feature detectors is acquired by using the detection operator at different scales.

The use of integral images allows the pyramidal scale space (see Appendix A.3) representation to be calculated by enlarging the box filters and convolving with the original image, a much faster process than consecutive blurring, downsampling and filtering. The scale space is divided into octaves, with each octave representing a doubling of scale. Each octave is divided into numerous scales to ensure scale invariance.

After applying the feature detector operation, features are localised by finding the maxima of the determinant and interpolating in scale and image space. Unfortunately, the rectangular nature of box filters cause them to maintain high frequency information that is traditionally lost over longer viewing distances, so scale invariance could be limited.

The SURF descriptor is similar to the SIFT descriptor in that it describes a keypoint by using image gradient information. Initially, Haar wavelet responses are calculated in the vertical and horizontal directions in a circular region of radius $6\sigma$ about the detected feature. Figure 3.12 shows the Haar wavelets. Note that no diagonal responses are considered, which could limit the descriptors rotational invariance.

The response strengths in the vertical and horizontal directions are used to select a dominant orientation for each feature patch by calculating the sum of all responses within a sliding window as indicated by Figure 3.13. The largest orientation vector over all sliding windows is then selected as the orientation of a feature. The SURF descriptor of each feature is measured relative to this orientation in order to obtain rotation invariance.

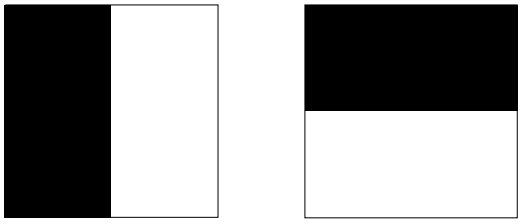


Figure 3.12: The figure shows the Haar wavelets used by SURF to estimate the gradient of regions around a detected feature.

Finally, the SURF descriptor is obtained by dividing a square region superimposed over the circular region used for dominant orientation assignment into 64 equal square regions. The square region is oriented in the direction of the dominant orientation. Each of the sub-regions within the oriented square is then assigned an orientation, but with the horizontal and vertical

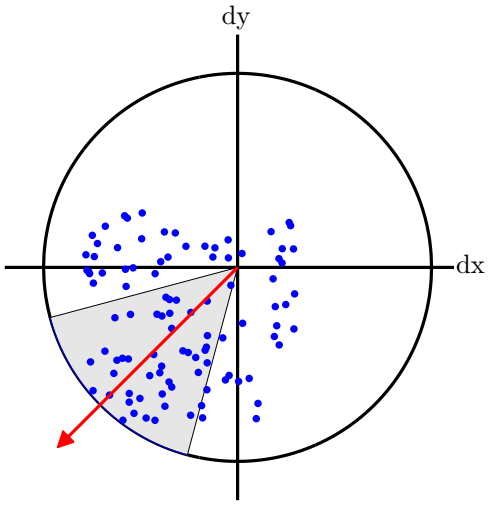



Figure 3.13: The response strengths in the vertical ($dy$) and horizontal ($dx$) directions are used to select a dominant orientation for each feature patch by calculating the sum of all responses within a sliding window and selecting the largest response over all the windows.

Haar wavelets rotated into the dominant orientation. Figure 3.14 shows the orientation-based descriptor for a hypothetical keypoint.

The descriptor is obtained by concatenating the four dimensional vector consisting of sums of the Haar wavelet responses and their absolute values for 16 larger regions around the keypoint.

In addition to the descriptor, the sign of the trace of the Hessian matrix approximation for each interest point is stored. Matching of feature points occurs by conducting a nearest neighbour search to find the most likely candidate descriptor match. The nearest neighbour search is sped up by testing that the signs of two potential interest point matches are the same, before computing the distance measure. A confirmed match is obtained if the signs of two interest points are the same and the distance measure between interest point descriptors falls within a certain threshold.

### 3.2.2 Semi-Naive Bayesian Classifier

The second feature detector of interest forgoes the descriptor matching approach and structures the matching as a semi-naive Bayesian classification problem. This approach, presented by Ozuysal *et al.* (2010), allows for keypoints to be matched without the intensive pre-processing requirements of the SURF detection scheme.

The problem is structured as follows. Given a set of image patches centered about various keypoints, and a new patch around a newly detected point, classify the new patch, using only a few binary tests, but assuming a classifier could be trained beforehand.

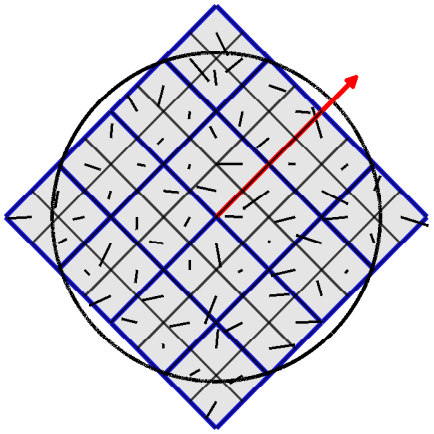


Figure 3.14: The SURF descriptor is obtained by finding the Haar gradient approximation in 64 regions of an image patch centered at the keypoint and oriented relative to the dominant keypoint orientation. These 64 regions are grouped into 16 larger regions, each containing 4 of the smaller areas. The sums of the horizontal and vertical gradient components, $\sum dx$ and $\sum dy$, together with the sums of their absolute values, $\sum |dx|$ and $\sum |dy|$ are calculated for each of these 16 regions. The 4 parameters from each of the 16 image patch regions are concatenated to form the 64 dimensional SURF descriptor.

Let $C_k$ be a class representing multiple views of an image patch about a particular keypoint. Furthermore, let us assume that $N$ binary feature tests can be conducted on each of the views of the image patch, and denote the results of these feature tests by $f_i$, where $i = 1, 2, \ldots, N$. Ozuysal *et al.* (2010) use the sign of the intensity difference between two randomly selected pixels in an image patch as the binary feature test.

Given a set of binary feature test results, $f_i$, of an image patch centered about a newly detected interest point, we need to select the class $k$ such that the conditional probability of these feature test results is maximised. Using Bayes rule, this implies that

$$k = \underset{k}{\operatorname{argmax}} \; P\left(C_k \,|\, f_1, f_2, \ldots, f_N\right) = \underset{k}{\operatorname{argmax}} \; P\left(f_1, f_2, \ldots, f_N \,|\, C_k\right), \quad (3.2.4)$$

assuming a uniform prior $P\left(C = C_k\right)$ and neglecting the scaling factor introduced by the denominator $P\left(f_1, f_2, \ldots, f_N\right)$.

Unfortunately, it is impractical to model the joint distribution of all the binary feature test results. Ozuysal *et al.* (2010) propose a compromise that collects $S$ binary features into $M$ groups of smaller size and assumes independence between these smaller sets. Each of these smaller sets is termed a fern, $F_j$, with $j = 1, \ldots, M$. The classification of this semi-naive Bayesian

formulation is then written as

$$k = \underset{k}{\operatorname{argmax}} \; P\left(f_1, f_2, \ldots, f_N \,|\, C_k\right) = \prod_{j=1}^{M} P\left(F_j \,|\, C_k\right). \tag{3.2.5}$$

#### 3.2.2.1 Training the Classifier

Each binary test on an image patch produces two possible results. This means that each fern, with $S$ features, will produce $2^S$ results, or a number between 0 and $2^S - 1$. By generating multiple images of the same patch and performing the binary feature tests, the output of the fern can be modelled as a multinomial distribution.

The distributions are initialised with a uniform Dirichlet prior, and training with a particular image updates the distributions for the corresponding class. Figure 3.15 shows fern training for a hypothetical image patch. After training, the binary test distributions over possible fern outputs are available for each class.

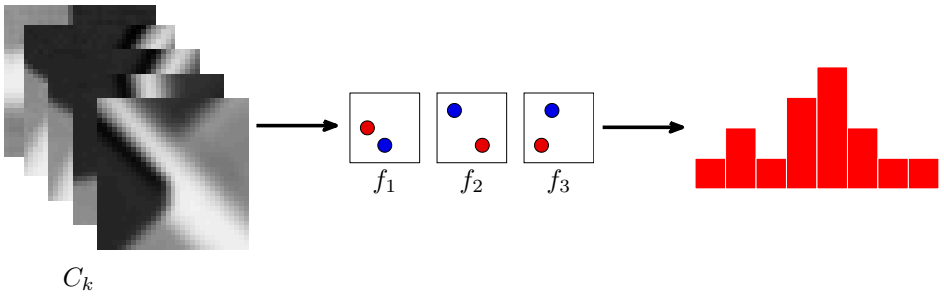



Figure 3.15: The training of a hypothetical fern. Assuming a fern is made up of three binary feature tests, there are eight possible results. Training the fern on multiple images of the same keypoint generates a multinomial distribution of the feature results. Each bin of the multinomial distribution corresponds to one of the eight possible binary test results.

#### 3.2.2.2 Classifying a New Patch

Figure 3.16 illustrates the classification process graphically for a hypothetical case, assuming a fern is made up of three binary feature tests and that there are five possible classes and three ferns. The distributions of each of the five classes are indicated in different colours.

Initially the binary tests are conducted for each fern (the image columns). The binary combination of test results for each fern points to a column of interest in the probability distribution of each class. The selected columns in each fern are multiplied to form the conditional probability of the image

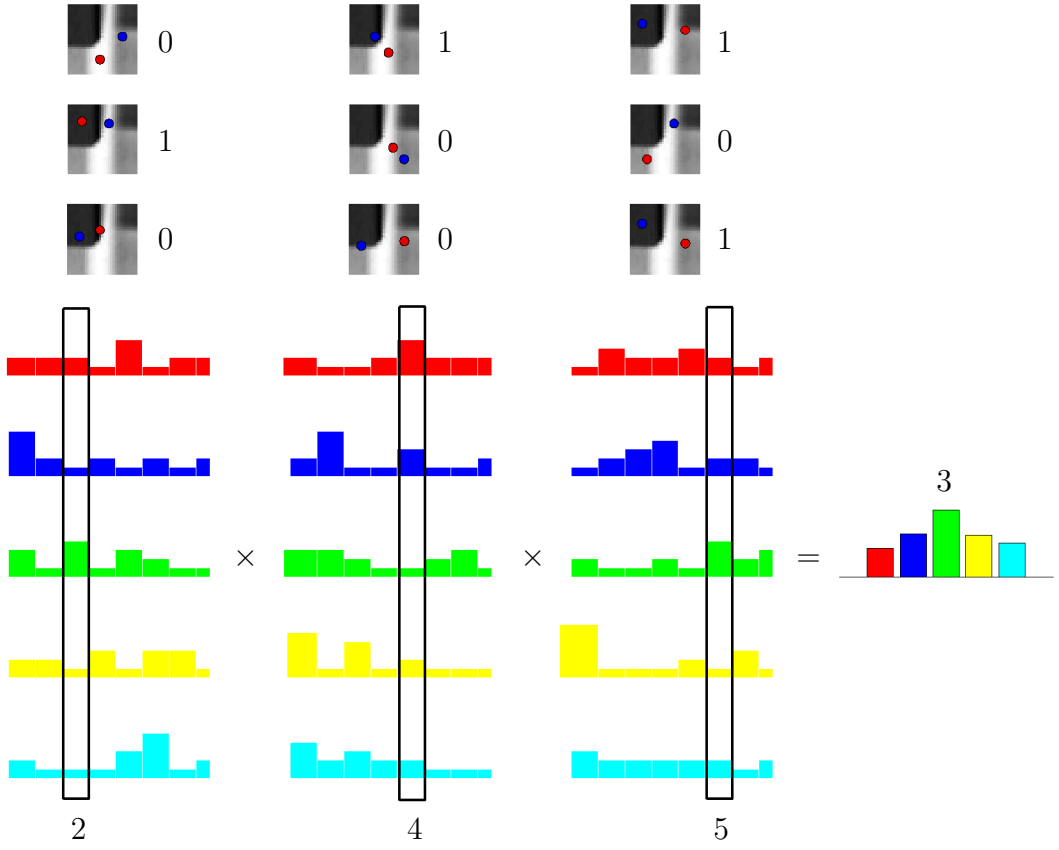



Figure 3.16: Given the binary test distributions, a new patch is easily recognised. The binary tests are conducted on the input patch, and the associated distribution column is selected for each class in every fern. The fern distributions are then multiplied to obtain the conditional probability of the image patch belonging to a class, and the image patch is judged to belong to the class corresponding to the largest point of the conditional probability distribution.

belonging to a class given the binary feature tests. The bin in which this distribution is maximised corresponds to the class to which the image patch belongs (in the illustrated example the image patch has been classified as belonging to class 3).

In practical use, far more tests, training images and ferns are used, while image patches are only classified if the conditional probability corresponding to the most likely class is above a certain threshold. Note that this classification approach requires multiple images of the patch around a keypoint for training. These images are generated by applying known affine, rotation and scale transformations to a single training patch. As a result, only a simple keypoint detector is required, as good keypoints are ensured by only using those which persevere across the transformed images.

### 3.2.3 Kanade-Lucas Feature Tracking

The final feature-based recognition strategy of interest is the Kanade-Lucas Tracker (KLT). The Kanade-Lucas tracker is a popular solution to the optical flow problem, estimating the apparent motion of features in a visual scene.

Assume $I(x, y; t)$ is the intensity of a keypoint in image $I$ at time $t$ and that the keypoint moves to a point $I(x + \delta_x, y + \delta_y; t + \delta_t)$ in a second scene over a small time step. The motion of the point can be approximated using a Taylor series expansion:

$$I(x + \delta_x, y + \delta_y; t + \delta_t) \approx I(x, y; t) + \frac{\partial I}{\partial x}\delta_x + \frac{\partial I}{\partial y}\delta_y + \frac{\partial I}{\partial t}\delta_t. \tag{3.2.6}$$

Let us assume that the intensity of the moving keypoint remains the same:

$$I(x, y; t) = I(x + \delta_x, y + \delta_y; t + \delta_t). \tag{3.2.7}$$

This is equivalent to assuming that lighting conditions remain stable and that glare and shadows are neglected. This means that

$$\frac{\partial I}{\partial x}\delta_x + \frac{\partial I}{\partial y}\delta_y + \frac{\partial I}{\partial t}\delta_t \approx 0. \tag{3.2.8}$$

Dividing through by $\delta_t$ gives

$$\frac{\partial I}{\partial x}\frac{\delta_x}{\delta_t} + \frac{\partial I}{\partial y}\frac{\delta_y}{\delta_t} + \frac{\partial I}{\partial t} \approx 0, \tag{3.2.9}$$

or

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} \approx 0, \tag{3.2.10}$$

with $V_x$ and $V_y$ the $x$ and $y$ components of the keypoint optical flow. Equation 3.2.10 has two unknowns and cannot be solved without incorporating additional constraints. Lucas and Kanade (1981) assume that the optical flow is locally consistent, or that the optical flow remains the same over a small region of size $m \times m$ about each keypoint.

This means that for each region, we can write $N = m^2$ equations with unknowns $V_x$ and $V_y$. This over-specified system can then be solved using a least squares approach to give

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum I_{x_i}^2 & \sum I_{x_i} I_{y_i} \\ \sum I_{x_i} I_{y_i} & \sum I_{y_i}^2 \end{bmatrix} \begin{bmatrix} -\sum I_{x_i} I_{t_i} \\ -\sum I_{y_i} I_{t_i} \end{bmatrix} \quad \text{for all } \; i \leq N. \tag{3.2.11}$$

Here, $I_{x_i}$ and $I_{y_i}$, are the image derivatives $\frac{\partial I}{\partial x}$ and $\frac{\partial I}{\partial y}$ in the horizontal and vertical directions respectively, renamed for simplicity. $I_{t_i}$ refers to the change in pixels over time, or the difference between pixel intensities in two frames. Typically, a Gaussian weighting function is used to add precedence to the optical flow of the pixel in the region centre.

Note that the KLT is primarily intended to operate on image streams, and not in the case where an input image is compared to a template image of a target captured in a completely different scene. As a result, the matching strategy discussed thus far, detecting an object in a scene by matching the scene with a reference image, is no longer suitable. Instead, the KLT needs to be used with some enrolment phase, where objects of interest are identified, and features detected are tracked over time.

### 3.2.4 Comparison of Feature Recognition Schemes

Results of the three feature detection schemes are now critically compared. Three sets of experiments were conducted, with each targeting a particular aspect or property required by a target-following system.

#### 3.2.4.1 Speed, Recognition Rates and Match Accuracy

The first test is conducted on a dataset of 320 images with a pixel resolution of $900 \times 680$ and aims to test the recognition rate, speed and match accuracy of each algorithm. The dataset consists of images of a moving planar target undergoing roll, pitch and yaw motions. The images were captured from a moving platform so as to better emulate the conditions of target following. In addition, the dataset contains images of target objects under partial occlusions, scale changes, as well as images where the target object is not present. Each image in the dataset was manually annotated with a Boolean value denoting the presence or absence of the target in the image scene.

Originally, results were obtained using various implementations of the feature matching or tracking algorithms, coded in both C++ and Matlab. However, in the interests of fair comparison, the decision was made to make use of the C++ Open Computer Vision Library (OpenCV) for all the experiments presented here. OpenCV, described in detail in the work of Bradski and Kaehler (2008), contains existing implementations of the SURF feature and descriptor extraction, the Ferns-based classifier and the KLT feature tracker. The experiments conducted here were performed on a desktop computer with a dual core 2.2 GHz processor and 2 GB memory.

Results were obtained by applying each of the three recognition strategies to each image in the dataset, using a common reference image as a template for matching. On processing each input image in the dataset, three sets of information were stored, whether the target was detected in an image, the locations of matched features in the image, and the time taken to perform this matching and extract this information. The SURF and KLT algorithms were supplied with a template image as training information, while the Ferns classifier was trained using warped images of the same template. The warped images were generated by applying random transformations encompassing roll, pitch and yaw rotations of up to $180^\circ$ and scale changes of up to 200%.

Table 3.1 shows the results of the first experiment comparing the three recognition strategies. The recognition rate is the percentage of correct decisions regarding the presence of the target in the dataset, with reference to the dataset labelling. Speed refers to the average frame rate achieved, while match accuracy refers to the ratio of clearly correct feature matches in the dataset, counted manually, to the total number of detected matches.

Table 3.1: Comparison of SURF, Ferns and KLT feature detectors.

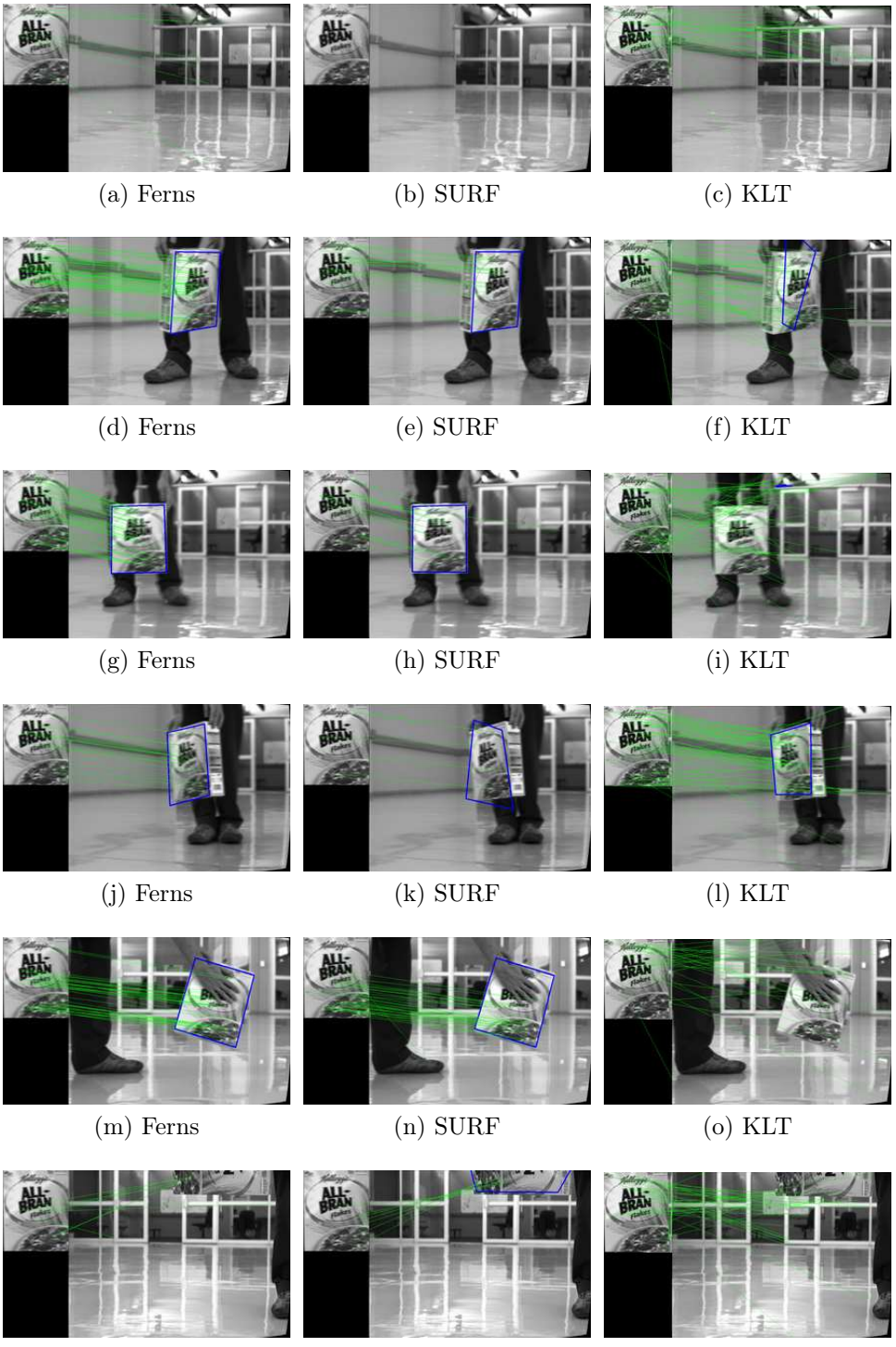
| **Feature Detector** | **Recognition Rate** (%) | **Speed** (fps) | **Match Accuracy** (%) |
|---|---|---|---|
| **SURF** | 94.08 | 1.54 | 98.83 |
| **Ferns** | 94.08 | 4.57 | 94.79 |
| **KLT** | 61.99 | 11.86 | 31.63 |

Both SURF and the Ferns approach resulted in extremely high recognition rates, while the KLT tracking was not as successful. It must be noted that the KLT tracker was by far the fastest algorithm, but its low recognition rate and match accuracy mean that it is not effective enough to be applied to the target-following problem. The performance of the KLT algorithm could be improved if it was re-initialised by either the SURF or Ferns algorithms, but this is beyond the scope of this work. The Ferns approach was a great deal faster than the SURF algorithm, but at the expense of more false matches, a factor indicated by the slightly lower match accuracy.

The higher number of false matches exhibited in the Ferns matching is most likely due to the classifier nature of the matching process. The SURF algorithm describes the keypoint surroundings in detail, but the Ferns algorithm merely conducts simple tests and selects a most likely match. This means that the SURF algorithm would produce keypoints better suited to pose estimation algorithms, as it appears to localise features more accurately.

Figures 3.17 and 3.18 show selected results obtained by processing images in the test set, using the three object detection algorithms. The template image is shown on the left of each image, with keypoint matches to the current scene marked by green lines. The blue quadrilaterals, superimposed on the figures by incorporating the homography transformation of Section 3.3, show the estimated projection of the template image onto the new scene.

Unfortunately, the superior matching accuracy of the SURF algorithm makes it somewhat less robust to the effects of motion blur. Figures 3.17g and 3.17h show how many more correct matches are found by the Ferns algorithm when processing a blurry image. This robustness of the Ferns approach causes it to frequently detect false positive matches, an example of which is provided in Figure 3.17a.



(a) Ferns (b) SURF (c) KLT

(d) Ferns (e) SURF (f) KLT

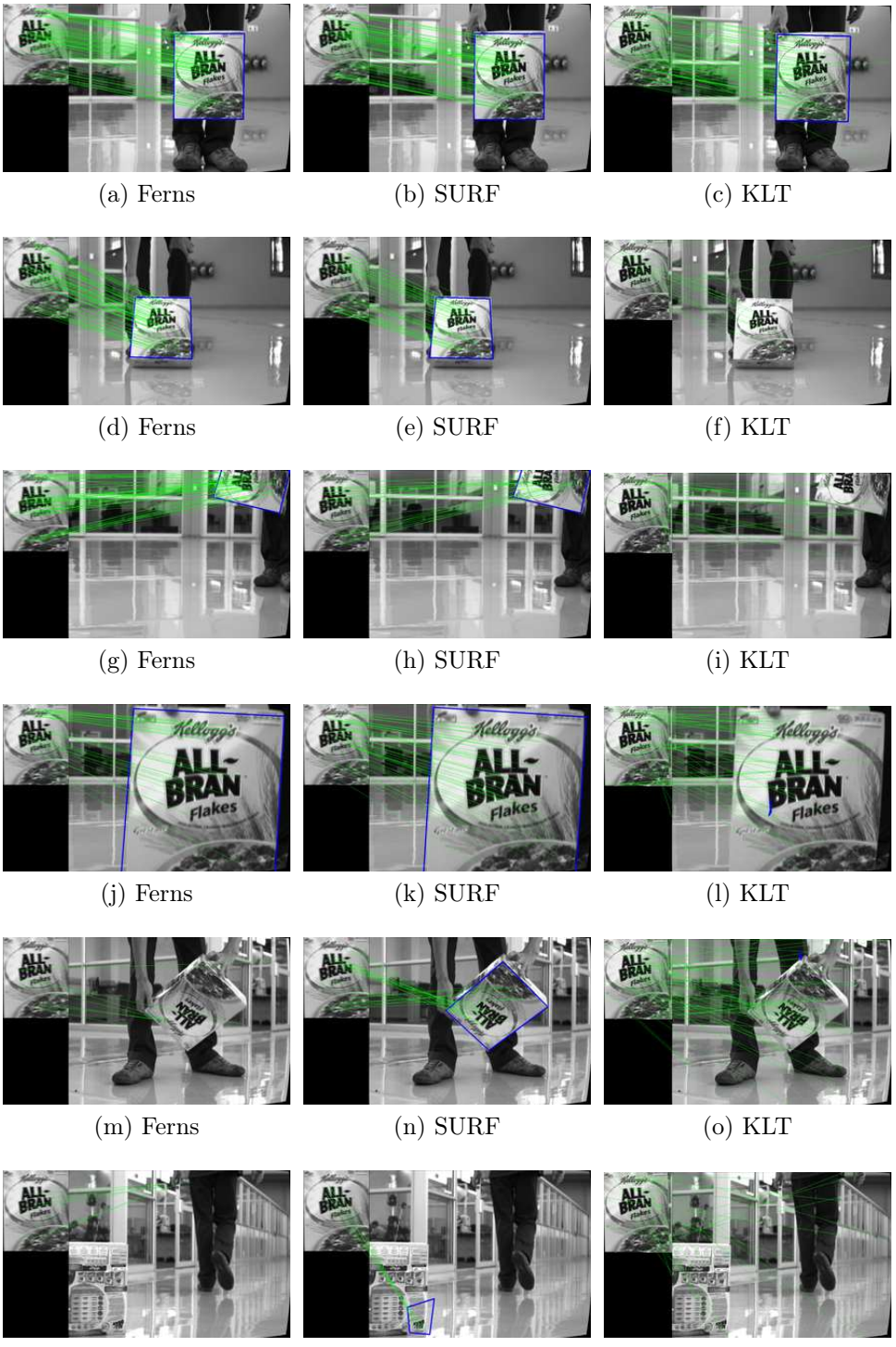
(g) Ferns (h) SURF (i) KLT

(j) Ferns (k) SURF (l) KLT

(m) Ferns (n) SURF (o) KLT

(p) Ferns (q) SURF (r) KLT

Figure 3.17: Selected feature matching results obtained by processing image scenes using the three object detection algorithms of interest.



(a) Ferns (b) SURF (c) KLT

(d) Ferns (e) SURF (f) KLT

(g) Ferns (h) SURF (i) KLT

(j) Ferns (k) SURF (l) KLT

(m) Ferns (n) SURF (o) KLT

(p) Ferns (q) SURF (r) KLT

Figure 3.18: Selected feature matching results obtained by processing image scenes using the three object detection algorithms of interest.

It should be noted that the SURF algorithm performed so well in one particular case that it was able to locate the logo on the rear of the cereal box used as a target, which contained only a small part of the template image. This case, displayed in Figure 3.18q, was marked as a false positive and negatively affected the recognition rate, as the human labelling the dataset did not consider the object to be present in this image. This particular case is useful however, as it illustrates the excellent appearance-based matching performance of the SURF algorithm.

The poor performance of the KLT algorithm is continually exhibited, with the numerous incorrect feature matches resulting in clearly incorrect projection estimates. Figures 3.17f and 3.18l show good examples of this non-ideal behaviour.

#### 3.2.4.2 Robustness to Yaw Motion

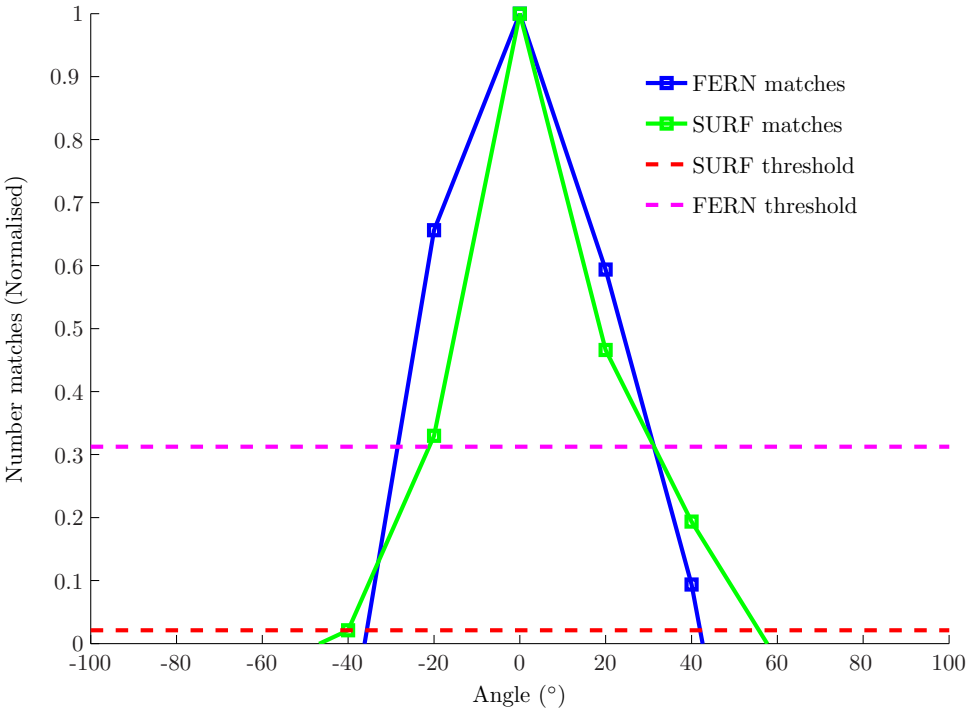
The second experiment tests the ability of the feature detectors to detect objects undergoing yaw motions. Only the SURF and Ferns algorithms are compared, since the first experiment clearly shows that the KLT is not suited to the target-following task. Images of a target object mounted on a turntable, at a fixed distance away from a camera, were captured at 20° intervals and processed using the two feature matching algorithms. Figure 3.19 shows the fraction of matches detected correctly over these yaw angles, which are likely to be encountered in a target-following operation.

Note that different thresholds are used to determine whether a match is made in each algorithm, so a compensation value is applied for the realistic comparison of algorithms. In addition, the number of matches detected are normalised. The figure shows that the Ferns algorithm typically produces more matches, and so is likely to be better performing, but does not operate over as wide a range of viewing angles as the SURF algorithm.

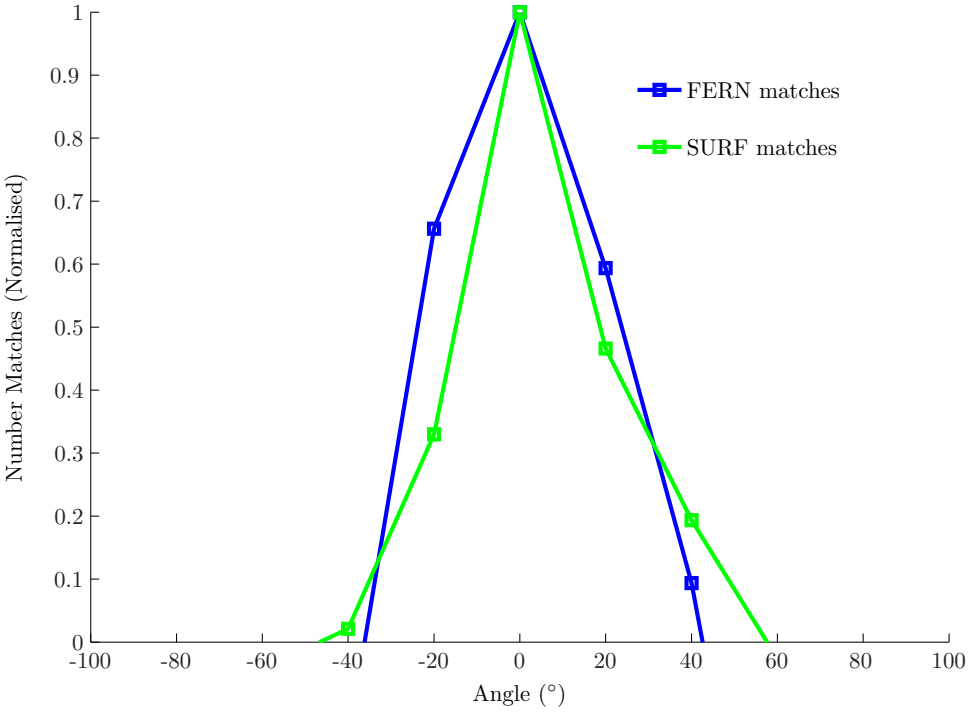
#### 3.2.4.3 Robustness to Scale Changes

Finally, the last experiment conducted aims to test the ability of the feature detectors to detect objects over larger scale changes. Once more, only the SURF and Ferns algorithms are compared, since the first experiment clearly shows that the KLT is not suited to the target-following task. The scaled images were generated by smoothing and downsampling a template image at scale intervals of 10%. Figure 3.20 shows the fraction of matches detected correctly over these scales.

Once more, a compensation value is applied for the realistic comparison of algorithms as different thresholds are used to determine whether a match is made for each algorithm. The figure shows that the Ferns algorithm typically produces more matches, and so is likely to be better performing, but does not operate over as large a range of scale changes as the SURF algorithm.

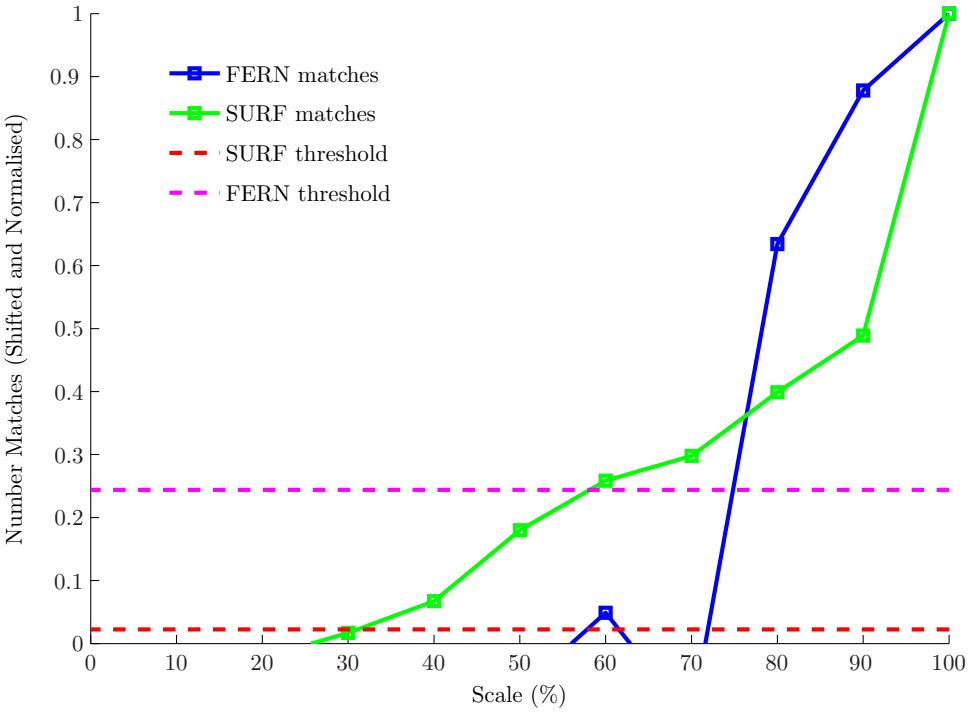



(a) Robustness to yaw (without threshold compensation)

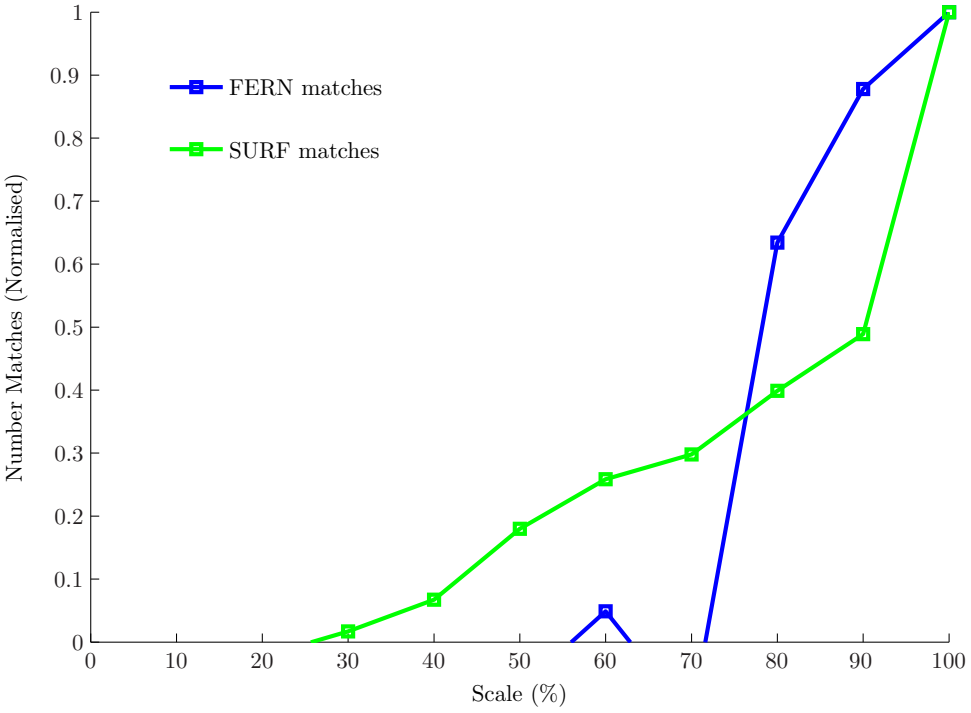



(b) Robustness to yaw (with threshold compensation)

Figure 3.19: The figure shows the system robustness to changes in yaw motion. Note that while the Ferns algorithm returns more matches in general, it does so over a lower range of yaw motions than the SURF algorithm.




(a) Robustness to scale (without threshold compensation)




(b) Robustness to scale (with threshold compensation)

Figure 3.20: The figure shows the system robustness to changes in scale. Note that while the Ferns algorithm returns more matches in general, it does so over a lower range of scale changes than the SURF algorithm. The Ferns algorithm was successful at a scale of 60%, but failed to obtain any matches at a 70% scale, presumably due to problems in the training of the classifier at this point.

#### 3.2.4.4 Selection of Feature Matching Scheme

The comparison of feature matching schemes above shows that the performance of the Ferns and SURF algorithms is similar at target viewpoints close to that of the template or reference image, with good matching accuracy and recognition rates. However, the SURF algorithm was shown to be more robust to changes in scale and yaw motion at extremities than the Ferns algorithm. The ability to detect targets over a wider range of viewpoints is more important that being more confident in detection over a smaller range of viewpoints, as it allows a target-following system greater leeway in navigation.

For this reason, the SURF algorithm is used in the human-following system. The better matching performance over changes in viewpoint does come at the expense of speed however, and various adjustments are required before the algorithm can be applied to a real-time situation.

### 3.2.5 Improving Matching Speed through Windowing

Though significantly faster than the SIFT algorithm on which it is based, the SURF matching scheme still requires improvements in speed to be of practical use. These improvements are obtained through the use of windowing and resolution adjustments.

Initially, the search for targets is conducted over an entire scene, at full camera resolution. Once detected, only the region of interest (ROI) in which the target is expected to appear is searched. This region of interest is obtained by expanding a window surrounding the detected object by $\frac{2}{3}$ of its detected width and height. Further improvements to speed are obtained by searching a downsampled image of this region of interest, should the target occupy significantly large regions of the image scene. Should the target be lost, the entire image is processed at full resolution and the process repeated.

This provides significant speed improvements, and allows the algorithm to process images at approximately 12 fps, under suitable lighting conditions. Even greater improvements to execution speed could be obtained if the algorithm was implemented on a GPU (graphical processing unit), but this is beyond the scope of this work.

## 3.3 Pose Estimation using Planar Objects

If the target followed is planar[2], all features on two views of it are related by the same homography, **H** (see (2.3.4)). This homography is obtained using the direct linear transform (DLT) described in Appendix C.1. A computationally efficient implementation of the normalised direct linear transform is included

[2]A planar target is one where all target points lie on the same plane in a 3D world coordinate system.

in the C++ Open Computer Vision library (Bradski and Kaehler (2008)) and used in this work.

Let $\mathbf{X}_1$ and $\mathbf{X}_2$ denote the 3D locations of two sets of co-planar points, with the point sets related by means of a rotation matrix $\mathbf{R}$, translation vector $\mathbf{t}$:

$$\mathbf{X_1} = \left(\mathbf{R} + \mathbf{t}\,\mathbf{n}^T\right)\mathbf{X_2}. \tag{3.3.1}$$

$\mathbf{n}$ denotes a vector normal to the co-planar points $\mathbf{X_2}$. If points $\mathbf{X}_1$ and $\mathbf{X_2}$ are viewed using a camera with intrinsic matrix $\mathbf{K}$, then the image plane projections of these points are denoted by:

$$\mathbf{x_1} = \mathbf{K}^{-1}\mathbf{X_1} \tag{3.3.2}$$

$$\mathbf{x_2} = \mathbf{K}^{-1}\mathbf{X_2}. \tag{3.3.3}$$

It can now be shown by substitution in (2.3.4) and solving for $\mathbf{H}$, that the homography relating the two image plane point sets can be decomposed as:

$$\mathbf{H} = \mathbf{K}\left(\mathbf{R} + \mathbf{t}\,\mathbf{n}^T\right)\mathbf{K}^{-1}. \tag{3.3.4}$$

Initially, the algorithm of Faugeras and Lustman (1988) was used to extract the rotation and translation relating two views of the detected object from the homography. This algorithm, described in Appendix C.2, is extremely lengthy and its derivation complex. An alternative to this algorithm can be used if the template or reference image is chosen intelligently.

If the template image is captured in a fronto-parallel position relative to the camera, it will have a surface normal $\mathbf{n} = [0,\ 0,\ 1]^{\mathrm{T}}$. Let $\widehat{\mathbf{H}} = \mathbf{K}^{-1}\,\mathbf{H}\,\mathbf{K}$ be the estimated homography with the intrinsic camera effects removed. Then

$$\widehat{\mathbf{H}} = \mathbf{R} + \begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{t}\end{bmatrix}, \tag{3.3.5}$$

where $\mathbf{t} = [t_x,\ t_y,\ t_z]^{\mathrm{T}}$ is the translation vector with components along the 3D Cartesian axes. If $\mathbf{r_1}$, $\mathbf{r_2}$ and $\mathbf{r_3}$ are the column vectors of the rotation matrix, with $\widehat{\mathbf{h}}_\mathbf{1}$, $\widehat{\mathbf{h}}_\mathbf{2}$ and $\widehat{\mathbf{h}}_\mathbf{3}$ the column vectors of the homography matrix with camera effects removed, then

$$\begin{bmatrix}\widehat{\mathbf{h}}_\mathbf{1} & \widehat{\mathbf{h}}_\mathbf{2} & \widehat{\mathbf{h}}_\mathbf{3}\end{bmatrix} = \begin{bmatrix}\mathbf{r_1} & \mathbf{r_2} & \mathbf{r_3}\end{bmatrix} + \begin{bmatrix}\mathbf{0} & \mathbf{0} & \mathbf{t}\end{bmatrix}. \tag{3.3.6}$$

This means that the first two columns of the rotation matrix are equal to those of the homography matrix and that the third column of the homography matrix contains both rotation and translation contributions. It is important to note that the first two column vectors of the homography matrix are orthonormal, as a result of the selected surface normal.

$$\mathbf{r_1} = \widehat{\mathbf{h}}_\mathbf{1} \tag{3.3.7}$$

$$\mathbf{r_2} = \widehat{\mathbf{h}}_\mathbf{2} \tag{3.3.8}$$

$$\mathbf{r_3} + \mathbf{t} = \widehat{\mathbf{h}}_\mathbf{3} \tag{3.3.9}$$

Recalling that the third column of a rotation matrix is equal to the cross product of the other columns, the translation vector is easily calculated.

$$\mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2} \tag{3.3.10}$$

$$\mathbf{t} = \widehat{\mathbf{h}}_\mathbf{3} - \mathbf{r_3} \tag{3.3.11}$$

The translation vector is returned up to scale, because a single camera is used. However, for the purposes of control, this ambiguity is not a problem as long as the translation components remain monotonic. A controller will minimise error in translation by generating proportional motion commands so, in a sense, the unknown scale is incorporated in the controller gains.

For the purposes of wheeled platform control, three parameters are of interest:

- $t_x$ : the horizontal translation between the template scene and currently viewed scene,
- $t_z$ : the depth or distance between the template scene and that currently viewed, along the optical axis of the camera,
- $\phi$ : the yaw, or target rotation around the vertical axis of the image plane.

These three parameters are termed the relative pose between the current and template scene. $\phi$ is extracted from the $3 \times 3$ rotation matrix by breaking it down into a roll, pitch and yaw configuration. It can then be shown that the yaw is calculated as

$$\phi = \text{atan}\left(\frac{r_{13}}{r_{33}}\right), \tag{3.3.12}$$

where $r_{13}$ is the element in the first row and third column of the rotation matrix. Similarly, $r_{33}$ is the element in the third row and third column of the rotation matrix.

Roll, pitch and vertical translation information is unnecessary and hence discarded. The ability to extract the three parameters of interest independently of the unnecessary degrees of freedom is important though, because it implies some invariance to uneven terrain.

## 3.4 Extension to Human Pose Estimation

The generalised target tracking system presented thus far needs to be extended before it can be applied to human following. In Section 2.5 the various types of wheeled robot control were introduced. They included point-to-point positioning, where robots moved directly to a point without considering orientation, and orientation inclusive or direction-based control, which causes the robot to move in such a way as to reach a point with a specific orientation. There are various benefits to each approach, and the primary goal of this work is to identify and compare these in the context of human following.

Before control algorithms that accomplish these manoeuvres can be implemented, the concept of human orientation needs to be explored. Let us assume that the pose of a walking person's upper body typically indicates travelling direction. Although humans are capable of walking in directions opposed to that indicated by their torsos, this is certainly not the norm, and intuitively, the assumption seems valid. The assumption is justified further by the work of Anderson and Pandy (2001).

Anderson and Pandy (2001) attempted to simulate human walking on level ground using a three-dimensional, neuromusculoskeletal model of the body together with dynamic optimisation theory. They compared their model with data captured from a variety of sources in human walking trials. This data showed that the deviation in back angle of a walking person typically remains within 10°.

As we are interested only in the approximate facing direction of the human, a complex model of body shape and limb position is not required. A simple planar fit to the back of the torso contains sufficient information for us to infer travelling direction.

Our goal is to find a means of fitting a plane through keypoints detected on the back of a human's torso. This is a relatively simple task if the 3D locations of features on the torso are known, but the only information available when a single perspective camera is used is the 2D locations of detected features on the image plane.

As discussed in Section 3.3, coplanar features in two views of an object are related by a homography $\mathbf{H}$. If we were to estimate the homography between two views of a human torso, we would effectively be measuring the rotation and translation between two planar regions of the torso.

Although features detected on the back of a human torso are usually not strictly coplanar, a sufficiently robust method of homography estimation is able to discard errors induced by this assumption. In addition, a robust measure of homography is also required to reduce errors caused by the deformable nature of clothing, which may ripple and warp during motion.

The problem of homography estimation is likely to be over-specified as typically more than four correspondences are found by the SURF matching scheme. Many correct matches would be useful in solving for the homography

in a least-squares sense, but incorrect matches (outliers) can have a drastically negative effect on such a solution. As a result, an iterative RANSAC-based approach (Fischler and Bolles (1987)) is used, in an effort to find a homography that minimises a re-projection error.

In this context RANSAC (short for random sample consensus) operates as follows. A random subset of four correspondences is drawn from the set of all available point correspondences and a model homography is determined using the direct linear transform. This homography is used with a re-projection error to determine which of the remaining points agree with the model, thereby forming a consensus set. In this case the re-projection error measures the error between the original coordinates of matched points and those projected in both directions under the model homography. If the consensus set is large enough the final homography is calculated from it as a least-squares solution. Alternatively, a new subset is chosen and the process is repeated until a large enough consensus set is obtained or a specified number of iterations is reached, in which case the final homography is calculated from the largest consensus set found.

This robust RANSAC-based homography estimation is extremely effective at obtaining homographies in the presence of a large number of outliers. This property is especially desirable as many outliers could be present in the human-following system due to the deformable nature of clothing, the occasional mismatched feature and the slight curvature (or deviation from planarity) of a human torso.

Note that the system requires that relatively salient clothing be worn by the human, because the detection is feature-based.

## 3.5 Tracking

After decomposition a pose vector $\mathbf{p} = [t_x, t_z, \phi]^{\mathrm{T}}$ is specified. It relates the current view that the robot has of the human with the desired view, as is illustrated in Figure 3.21.

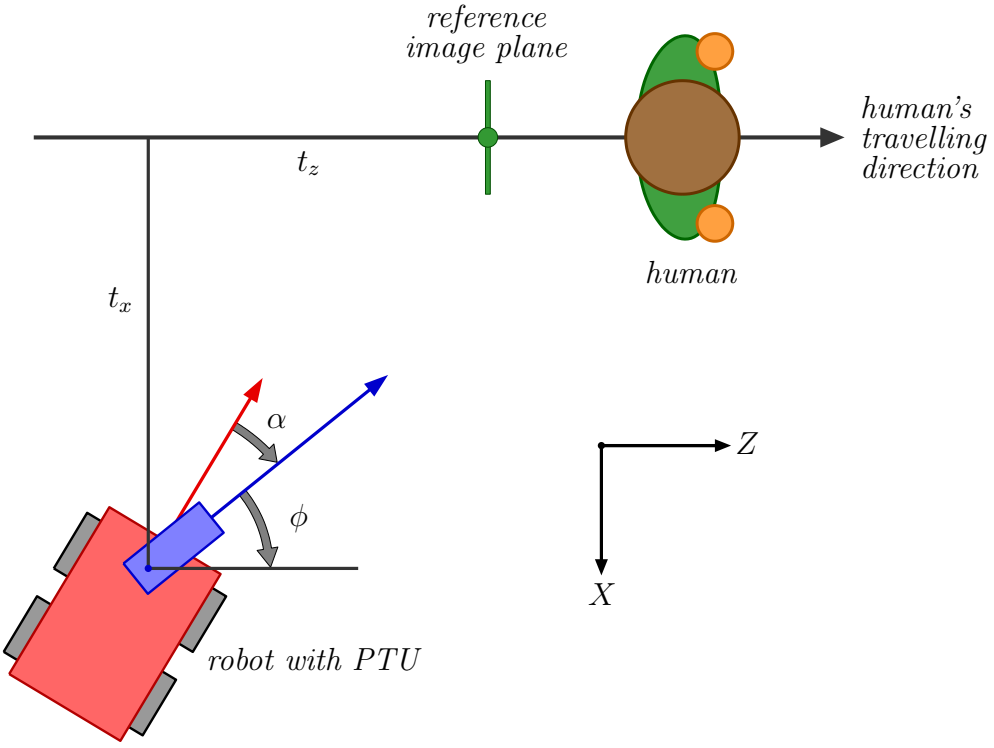



Figure 3.21: A graphical interpretation of the parameters $t_x$, $t_z$ and $\phi$ that specify the pose vector relating the robot's position and orientation with the human's. The parameter $\alpha$ represents the relative angle between the camera and platform.

$t_x$ represents the horizontal translation component between the two camera views and $t_z$ the translation component along the initial camera's optical axis. The angle $\phi$ represents the camera yaw, or angle between the human facing direction and the optical axis of the current camera, and is extracted from the rotation matrix $\mathbf{R}$ through the angle-axis parametrization of a roll, pitch and yaw motion.

The pose measurements could lose accuracy as the target object nears the extremal regions over which detection is successful. As a result the pose estimate needs to be refined further before it can be used in a control algorithm.

This is accomplished through the use of an extended Kalman filter (EKF). The extended Kalman filter is discussed in detail in Appendix D.1.

The uncertainty, or variance $\mathbf{R_k}$, in the pose measurement is selected as one third of the typical pose variation in straight line motion, weighted by a factor $w = 1 - \frac{n_i}{n_t}$. Here $n_i$ indicates the number of inliers or size of the consensus set, used to estimate the homography, and $n_t$ is the total number of available matches. The weighting implies that as the size of the consensus set increases so does the trust in the estimated homography and its decomposition.

An approach similar to that of Yoon *et al.* (2008) is followed to predict pose from a history of estimates in previous frames. Yoon *et al.* (2008) use this prediction to refine an estimate of the six degrees of freedom in a rigid object's pose. Let $\mathbf{p}_{k-1|k-1}$ be a pose estimate at time index $k-1$. The predicted pose at time step $k$ is then given by

$$\mathbf{p}_{k|k-1} = \left(1 + \frac{\delta t_k}{\delta t_{k-1}}\right) \mathbf{p}_{k-1|k-1} - \left(\frac{\delta t_k}{\delta t_{k-1}}\right) \mathbf{p}_{k-2|k-2} + \zeta_k, \tag{3.5.1}$$

where $\delta t_k = t_k - t_{k-1}$ is the time elapsed between steps $k-1$ and $k$, and $\zeta_k$ is the zero-mean prediction noise, with covariance $\mathbf{Q}_k$, associated with time step $k$. This simple model assumes that the time rate of pose change remains constant. Under the assumption that the prediction noise is independent of pose estimates, the predicted covariance of $\mathbf{p}_k$ can be written as follows:

$$\mathbf{P}_{k|k-1} = \left(1 + \frac{\delta t_k}{\delta t_{k-1}}\right)^2 \mathbf{P}_{k-1|k-1} + \left(\frac{\delta t_k}{\delta t_{k-1}}\right)^2 \mathbf{P}_{k-2|k-2} + \mathbf{Q}_k. \tag{3.5.2}$$

The uncertainty in predicted pose, $\mathbf{Q}_k$, is selected as half the uncertainty in measurement (excluding the homography trust weighting) so as to stabilise the measurements based on their history. This produces action similar to that of an exponentially weighted filter.

The measurement model is assumed to be unity, as the pose is measured directly. This means that the measurement and covariance residuals are:

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{p}_{k|k-1} \tag{3.5.3}$$

$$\mathbf{S}_k = \mathbf{P}_{k|k-1} + \mathbf{R}_k. \tag{3.5.4}$$

Then, the updated state and covariance estimate is given by:

$$\mathbf{p}_{k|k} = \mathbf{p}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \tag{3.5.5}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k)\mathbf{P}_{k|k-1}, \tag{3.5.6}$$

with $\mathbf{K}_k = \mathbf{P}_{k|k-1}\mathbf{S}_k^{-1}$ the optimal Kalman gain for a linear system.

Note that only a simple motion model is used to predict the future behaviour of the human target. An improved estimate could be obtained if a better model of human motion was incorporated, but the design and definition of a suitable model of human motion is an extremely challenging topic in itself, and thus not addressed in this work.

## 3.6 Control

The design of the human-following controllers is now presented. Three controller types are discussed and compared in this thesis, a point-to-point positioner, a direction-based controller and a hybrid controller attempting to combine the benefits of both. As discussed previously, the goal of this work is to examine the benefits of direction-based control over point-to-point positioning in the context of human following and the human orientation measure introduced in Section 3.4.

The objective in direction-based control is to generate motion commands that cause the current view of the target to resemble that of a template or reference image. Point-to-point control, on the other hand, merely tries to generate motion commands that maintain a fixed distance from the target. As discussed previously, it is not possible to extract absolute scale when using a single perspective camera. However, control systems can be designed using the extracted structure that does not include scale. In order to simplify this, the assumption is made that the fronto-parallel template image is captured from the desired platform position, measured relative to the target.

The human-following system is implemented on a terrestrial skid steering platform with two available control variables: angular and forward velocity. Before the controllers are presented, however, one of the fundamental problems of position-based visual servo control when used with a platform that has highly constrained motion needs to be addressed. Assuming an eye-in-hand configuration, one where the camera is fixed to the mobile platform, following a target may require a trajectory that would result in sight of the target being lost. Figure 3.22 illustrates this problem.

The problem is remedied by mounting the camera on a pan-tilt unit (PTU). This essentially adds a degree of freedom in order to facilitate the control process and allow certain trajectories to be followed without losing sight of the target. The pan angle of the PTU is of interest for motion control and is denoted by $\alpha$. Figure 3.21 shows this angle, which measures the difference between the camera's facing direction and the front of the platform.

### 3.6.1 Pan-Tilt Unit Control

The PTU used in this work is a commercially available unit manufactured by FLIR Motion Control Systems, formally known as Directed Perception. The PTU responds to RS232 serial commands for pan and tilt position, and has adjustable speed settings. The speed of PTU movement needs to be set with care. Ideally, the PTU should move to a desired position as fast as possible, or at the least much faster that the highest frequency of platform controller operation. Unfortunately, moving the PTU too quickly results in camera motion blur, which severely hampers the detection and matching of features.

The PTU is controlled in such a way as to always point the camera towards the centroid of detected features. The pan and tilt angles required to point the camera at the target object centroid are calculated as follows.

If $s$ is the size of pixels on the camera CCD (charge coupled device), and $f$ the lens focal distance in millimetres, the pan angle between a detected point and the image centre is calculated as:

$$\alpha = \text{atan}\left(\frac{s\,(c_x - x_c)}{f}\right). \tag{3.6.1}$$

$c_x$ is the horizontal image coordinate of the detected centroid and $x_c$ the horizontal image coordinate of the image centre.

Similarly, the tilt angle $\beta$ is calculated as:

$$\beta = \text{atan}\left(\frac{s\,(c_y - y_c)}{f}\right), \tag{3.6.2}$$

with $c_y$ the vertical image coordinate of the detected centroid and $y_c$ the vertical image coordinate of the image centre.

The pixel size $s$ is usually specified by the camera manufacturer, while the focal distance $f$ is a property of the lens, also typically provided by the lens manufacturers. In this work, a Prosilica GigE Ethernet camera with 7.5 $\mu$m size pixels is used together with a 12.5 mm lens.

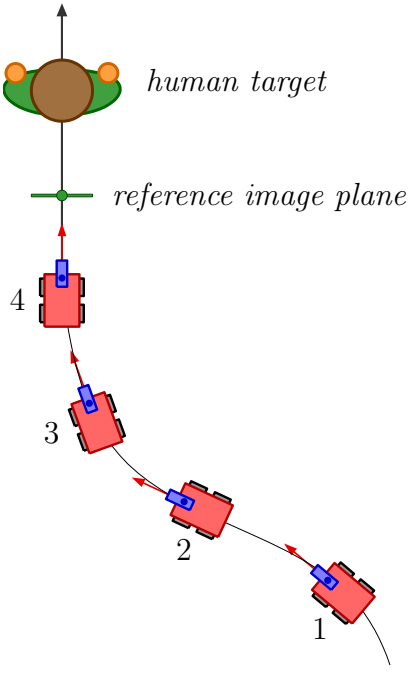



Figure 3.22: A potential trajectory generated through direction-based control. At position 2, sight of the target is lost and the following task would fail.

The PTU continually adjusts its azimuth and elevation so that the centroid of matched features is centred in the camera frame. From this point on, the assumption that the target is centred in the camera view is made.

### 3.6.2 Process Model

The three controllers of interest in this thesis were designed using a linearised model of the robot and vision system. This model is discussed and presented here. Figure 3.23 shows the measured parameters with which the robot is controlled. The figure is repeated so as to facilitate understanding of the control algorithms presented here.

The model presented here uses a kinematic unicycle motion model (see Section 2.5) to describe the motion of the robotic platform. This model assumes that the dynamics of the platform are negligible. Similarly, it is assumed that the PTU controller operates significantly faster than the platform, and the dynamics of the PTU are neglected. Assuming that the PTU controller causes the camera to face the target at all times, the discretised platform model is as follows:

$$\begin{aligned} t_x(k) &= t_x\left(k-1\right) - v_{k-1}\sin\left(\theta\left(k-1\right)\right)\Delta_t \\ t_z(k) &= t_z\left(k-1\right) - v_{k-1}\cos\left(\theta\left(k-1\right)\right)\Delta_t \\ \theta(k) &= \theta\left(k-1\right) + \omega_{k-1}\Delta_t \\ \phi(k) &= \operatorname{atan}\left(\frac{t_x(k)}{t_z(k)}\right) \\ \alpha(k) &= \theta(k) - \phi(k). \end{aligned} \tag{3.6.3}$$

Here, $\theta(k)$ represents the orientation of the platform. Note that $\theta(k)$ is equivalent to the sum of $\alpha(k)$ and $\phi(k)$. $k$ denotes the current sample, with $v$ and $\omega$ representing the forward and rotational velocities of the platform.

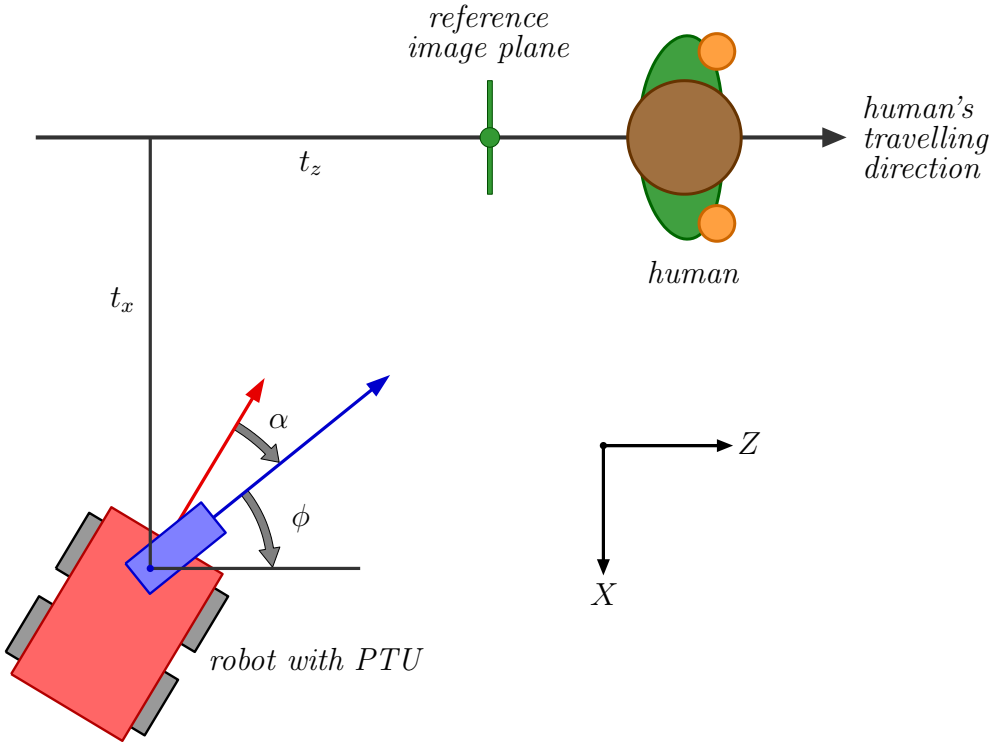



Figure 3.23: A graphical interpretation of the parameters $t_x$, $t_z$ and $\phi$ that specify the pose vector relating the robot's position and orientation with the human's. The parameter $\alpha$ represents the relative angle between the camera and platform.

It is important to note that while the translations $t_x$ and $t_z$ are measured up to a time-invariant scale in practice, the model presented here does not take this into account. As a result, all control gains need to be adjusted to accommodate this unknown scale when control systems are practically implemented.

The model of (3.6.3) needs to be linearised before it can be used to design suitable control systems. Using Taylor series expansion to expand these equations, and discarding higher order terms provides a linear model,

$$\begin{aligned} t_x(k) &= t_x(k-1) - 0.75\theta(k-1)\Delta_t \\ t_z(k) &= t_z(k-1) - v_{k-1}\Delta_t \\ \theta(k) &= \theta(k-1) + \omega_{k-1}\Delta_t \\ \phi(k) &= t_x(k) \\ \alpha(k) &= \theta(k) - \phi(k). \end{aligned} \tag{3.6.4}$$

The model was linearised about an operating point of $t_x = 0\,m$, $t_z = 1\,m$, $\theta = 0°$, $v = 0.75\,m/s$ and $\omega = 0\,rad/s$. This point was selected as this

represents the desired configuration between the human target and following robot. It is assumed that the human travelling velocity will not exceed $1.5\,m/s$ and for this reason a midpoint forward velocity of $0.75\,m/s$ was selected for the linearisation.

The plant model was transformed to the z-domain for the purposes of digital control design, using a sampling frequency of 12 fps. This leads to the transfer function,

$$\mathbf{G}(z) = \begin{bmatrix} 0 & -\frac{0.75\Delta_t^2}{(1-z^{-1})^2} \\ -\frac{\Delta_t}{1-z^{-1}} & 0 \\ 0 & \frac{\Delta_t}{1-z^{-1}} \\ 0 & -\frac{0.75\Delta_t^2}{(1-z^{-1})^2} \\ 0 & \frac{\Delta_t}{1-z^{-1}} + \frac{0.75\Delta_t^2}{(1-z^{-1})^2} \end{bmatrix} \tag{3.6.5}$$

where $\Delta_t$ is the sampling time and

$$\begin{bmatrix} t_x(z) \\ t_z(z) \\ \theta(z) \\ \phi(z) \\ \alpha(z) \end{bmatrix} = \mathbf{G} \begin{bmatrix} v(z) \\ \omega(z) \end{bmatrix}. \tag{3.6.6}$$

The following sections use this model to select control parameters for each of the control systems of interest in this thesis.

### 3.6.3 Point-to-Point Positioning

The first controller presented is a point-to-point positioner. This controller causes the platform to move towards a desired point, with orientation uncontrolled. The use of a position-based visual servo control strategy allows for simple, decoupled control laws to be implemented. Although the PTU controls both pan and tilt angles, only the pan angle is of interest for platform motion control.

The forward velocity control law adjusts the straight line speed of the platform, based on the translational difference between the current viewpoint and the point at which the reference image was captured, along an axis perpendicular to the reference image. The forward velocity control input, $v(k)$, is obtained through the proportional-integral control law:

$$v(k) = K_1 \left[ t_z(k) + \tau_i \sum_{m=0}^{k-1} t_z(m) \right], \tag{3.6.7}$$

with $t_z(k)$ the vertical translation component of the extracted pose measurement at time $k$. Here $K_1$ is a proportional gain and $\tau_i$ an integral term that

rejects cumulative errors introduced by a moving target. As this control system operates in real time, a minimum acceptable sampling rate is required for the system to remain stable. The integral term is added at the expense of phase lag, which increases the required processing rate and necessitates high-speed image processing. A proportional-integral control law was selected so as to ensure zero steady-state error to a ramp input, given the first order integrator between $v(z)$ and $t_z(z)$,

$$\mathbf{G}_v = -\frac{\Delta_t}{1 - z^{-1}}. \tag{3.6.8}$$

Figure 3.26 shows the controller and control architecture used for forward velocity control. The parameter KTs in the integrator block is equivalent to the sampling time $\Delta_t = 1/12\,s$.

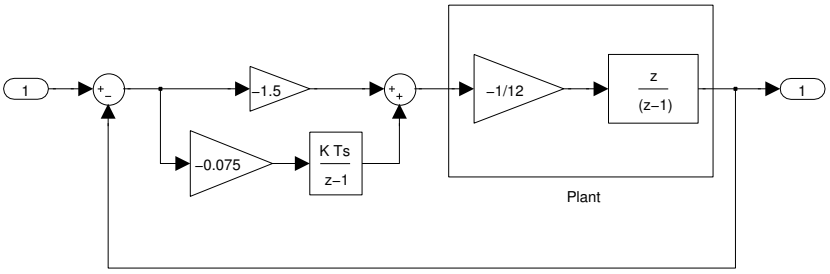



Figure 3.24: Control architecture used for forward velocity control.

The parameters of interest in this control law were selected using pole placement and root locus[3] design techniques. Figure 3.25 shows the root locus of the open loop transfer function incorporating the velocity controller. Note that the forward velocity control loop of the plant has a pole[4] at $z = 1$ and a zero[5] at $z = 0$. The proportional-integral controller introduces a pole at $z = 1$ and a zero was placed at $z = 0.9958$.

Controller gains and the location of the zero was selected so as to ensure a system bandwidth of $1.4\,rad/s$, significantly slower than the average image processing frame rate. This results in a phase margin of 91.7° and ensures controller stability. This is confirmed by the stable closed loop pole at $z = 0.893$. The closed loop system has a zero at $z = 0$. Care was taken to ensure that the forward velocity control does not exceed the platform maximum of $2\,m/s$.

[3]A root locus is a graphical visualisation of the motion of system roots as a function of an adjustable parameter such as controller gain.

[4]A pole is a point at which a function has a singularity. The poles of a system are used to examine its stability (in this context stability refers to the ability of system to produce a bounded output in response to a bounded input). An unstable system is one where the poles fall outside a unit circle in the z-plane.

[5]A zero is a point at which a function loses rank. Zeros affect the controllabilty of a system.

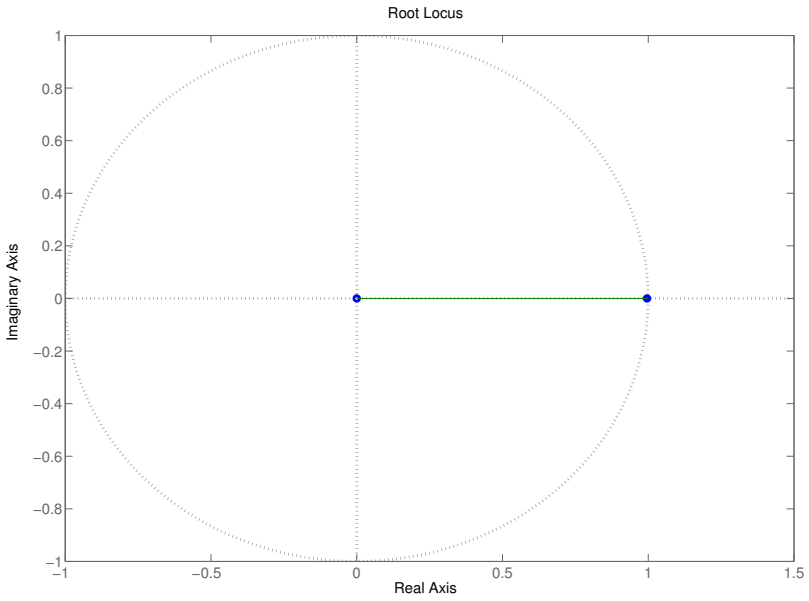



Figure 3.25: Open loop root locus for forward velocity control.

The platform's angular velocity input at time step $k$ is generated by the proportional control law

$$\omega_1(k) = K_2\,\alpha(k). \tag{3.6.9}$$

Assuming the PTU controller keeps the camera pointed at the target centre, adjusting the rotational speed proportionally to the pan angle, $\alpha$, ensures that the platform always points towards the target. Only a proportional controller is used here to provide zero steady-state error, as the transfer function between $\omega(z)$ and $\alpha(z)$,

$$\mathbf{G}_{\omega_1} = \frac{\Delta_t}{1 - z^{-1}} + \frac{0.75\Delta_t^2}{(1 - z^{-1})^2}, \tag{3.6.10}$$

is fourth order and contains an integrator. The open loop transfer function has two poles at $z = 1$, with zeros at $z = 0$ and two more at $z = 0.9412$. Figure 3.26 shows the controller and control architecture used for rotational velocity control.

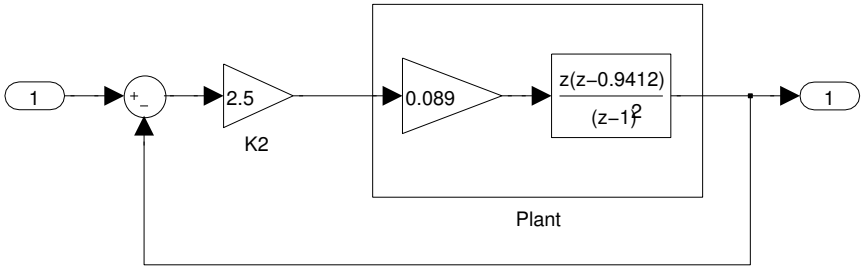



Figure 3.26: Control architecture used for rotational velocity control.

Once more, gains were selected using root locus design. Figure 3.27 shows the root locus of the open loop transfer function incorporating the rotational velocity controller.

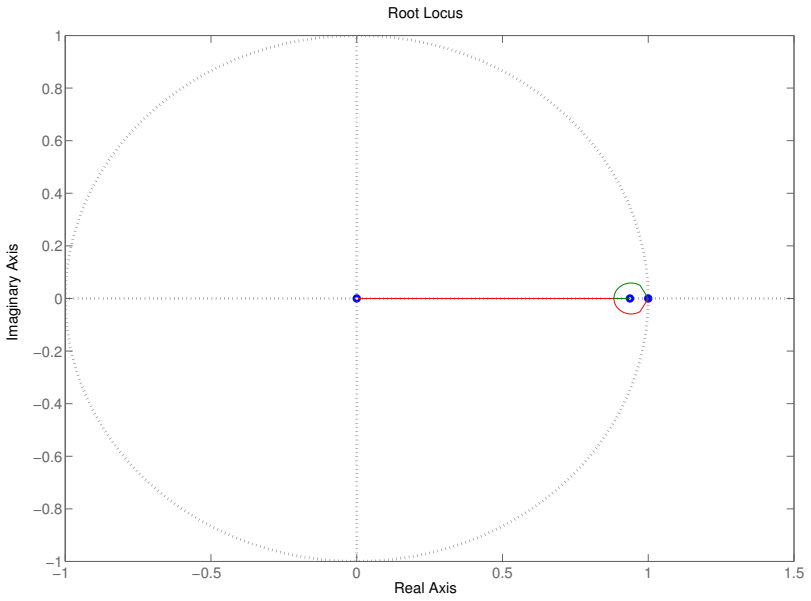
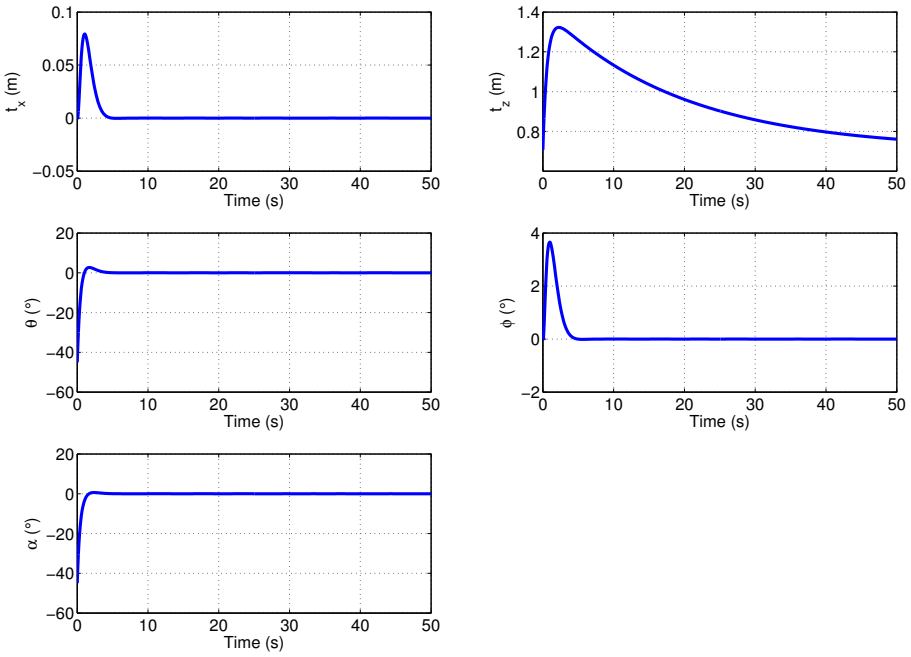


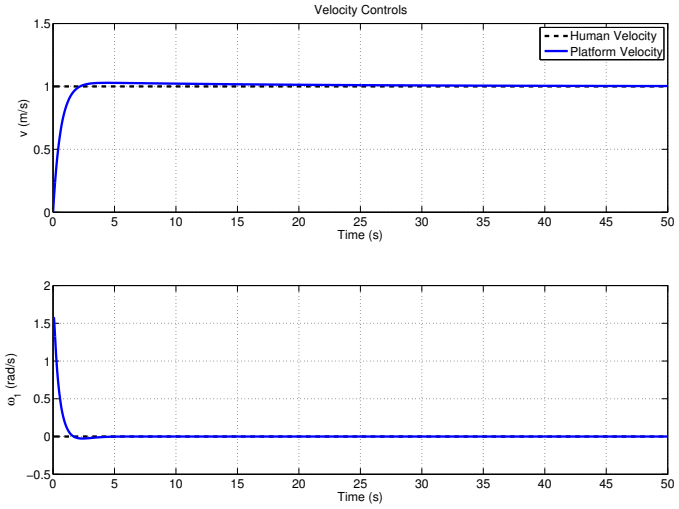

Figure 3.27: Open loop root locus for rotational velocity control.

The gain was adjusted so as to achieve a system bandwidth of approximately $2.87\,rad/s$, significantly lower than the average image processing rate, resulting in a phase margin of 79.1°. The closed loop system is underdamped with poles at $z = 0.0908 \pm 0.0488\,\mathrm{i}$, providing a damping factor of 0.87. This damping factor was selected so as to ensure that the controlled rotational velocity does not exceed $\frac{\pi}{2}\,rad/s$. This reduces the risk of motion blur associated with a rapidly rotating camera and keeps the rotational velocity within the limits of the platform. The closed loop has a zero at $z = 0.9375$.

Figure 3.28 shows the disturbance response of the coupled system to a target offset by $t_x = 0.5\,m$, $t_z = 0.5\,m$ and $\alpha = 45°$, when the target moves forward in a straight line at $1\,m/s$.




(a) Plant response




(b) Controls

Figure 3.28: Disturbance response of system to a step change in an offset targets velocity when using point-to-point control.

The crosstrack error $t_x$ and relative orientation $\phi$ eventually converge to zero in this case, even though these parameters are not actually controlled. While the point-to-point controller follows the target's position closely, it does

not take orientation into account. As a result it may be vulnerable to losing a sharply turning target.

Table 3.2 shows the gains and time constants used in the point-to-point control scheme.

Table 3.2: Point-to-point controller parameters.

| Parameter | Value |
|---|---|
| $K_1$ | $-1.5$ |
| $\tau_i$ | $-0.075$ |
| $K_2$ | 2.5 |

### 3.6.4 Direction-based Control

The second controller presented regulates both position and orientation. The forward velocity control input remains the same as that of the point-to-point positioner, listed in (3.6.7).

The platform's angular velocity input at time $k$ is generated by the proportional control law

$$\omega_2(k) = K_3\big[\phi(k) + \alpha(k)\big] + K_4\, t_x(k). \tag{3.6.11}$$

Here $\phi(k)$ is the estimated human facing direction extracted from the homography, $\alpha(k)$ the PTU pan angle and $t_x(k)$ the cross track error at time step $k$. The cross track error is defined by the signed magnitude of the perpendicular line segment between the reference trajectory and the robot. In our case the reference trajectory is the line passing through the desired camera centre, oriented in the direction of its optical axis (see Figure 3.23). The cross track error is therefore equivalent to the horizontal translation component of the extracted pose measurement.

The proportional gains $K_3$ and $K_4$ in Equation 3.6.11 are used to weight the relative errors in the control law. In general these errors cannot be minimised simultaneously as they typically represent conflicting goals, so the weighting needs to be selected such that greater emphasis is placed on minimising cross track error.

The open loop transfer function for the rotational velocity loop is extracted from (3.6.5) and given by

$$\mathbf{G}_{\omega_2} = \begin{bmatrix} -\frac{0.75\Delta_t^2}{(1-z^{-1})^2} \\ \frac{\Delta_t}{1-z^{-1}} \end{bmatrix}, \tag{3.6.12}$$

where

$$\begin{bmatrix} t_x(z) \\ \phi(z) + \alpha(z) \end{bmatrix} = \mathbf{G}_{\omega_2}\omega(z). \tag{3.6.13}$$

The plant has a pole at $z = -1$ and a zero at $z = 0$. Note that the cross track error $t_x(z)$ is related to the integral of $\phi(z) + \alpha(z)$. This allows the system to be treated as a single-input single-output system with two feedback control loops, when using the control architecture of Figure 3.29.

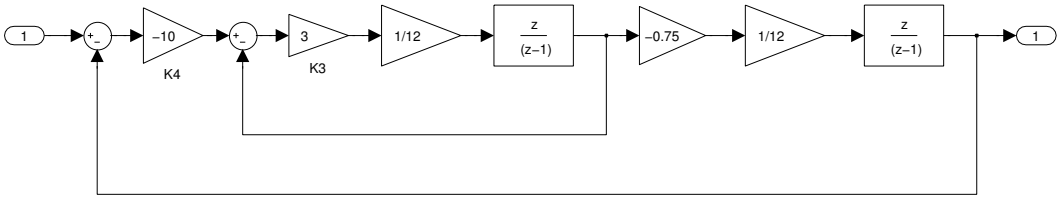



Figure 3.29: Architecture of direction-based orientation controller.

This control architecture allows for each of the gains in the direction-based control law to be selected independently, by designing each of the control loops separately.

Figure 3.30 shows the open loop root locus of the inner loop corresponding to the orientation regulation component of the direction-based controller.

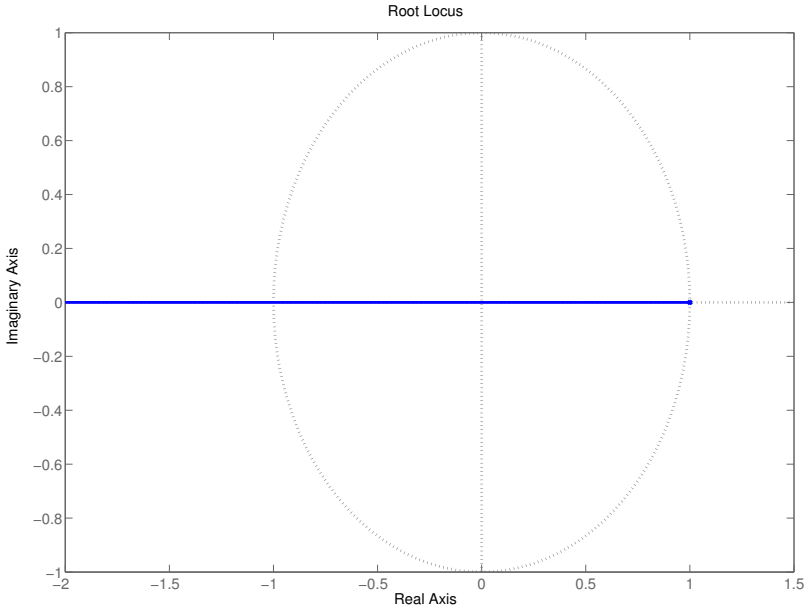



Figure 3.30: Open loop root locus for orientation regulation.

The proportional gain $K_3$ was selected so that the system bandwidth was approximately $0.25\,rad/s$. This is significantly lower than the bandwidth designed for in the point-to-point control scheme. This bandwidth was selected because minimising crosstrack error is more important in the direction-based

control scheme, and so the response of the orientation regulation loop needs to be slower than that of the loop regulating cross track error.

The controlled inner loop is closed loop stable, with a pole at $z = 0.9792$ and a phase margin of 89.4°.

Figure 3.31 shows the open loop root locus of the outer loop corresponding to the cross track error regulation component of the direction-based controller, given the controlled inner loop transfer function.

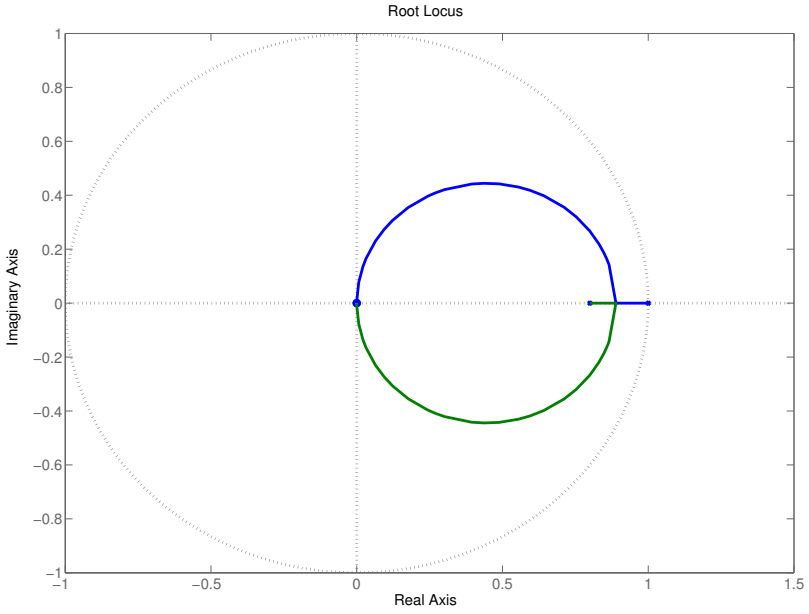
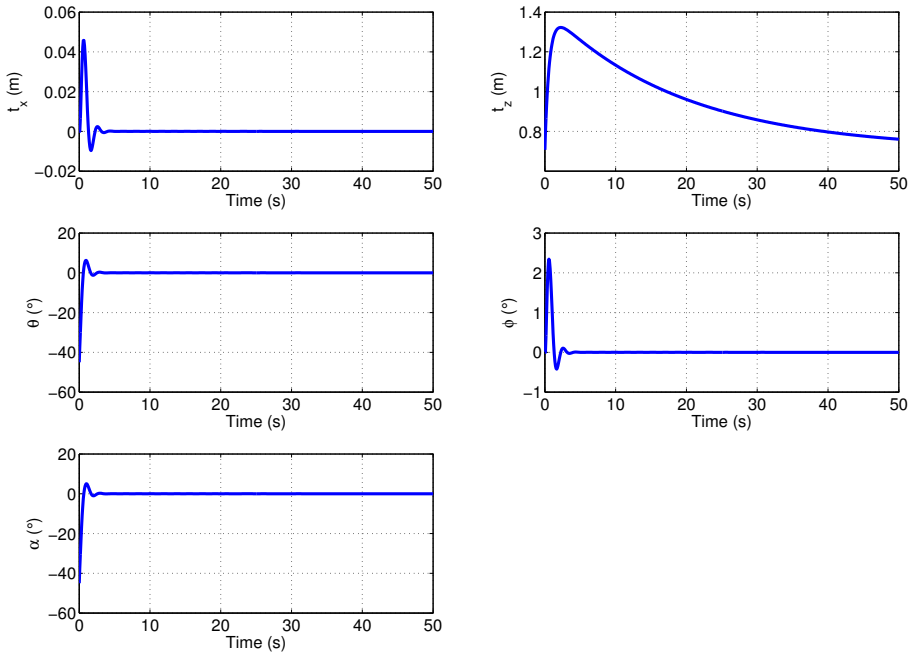



Figure 3.31: Open loop root locus for cross track regulation.

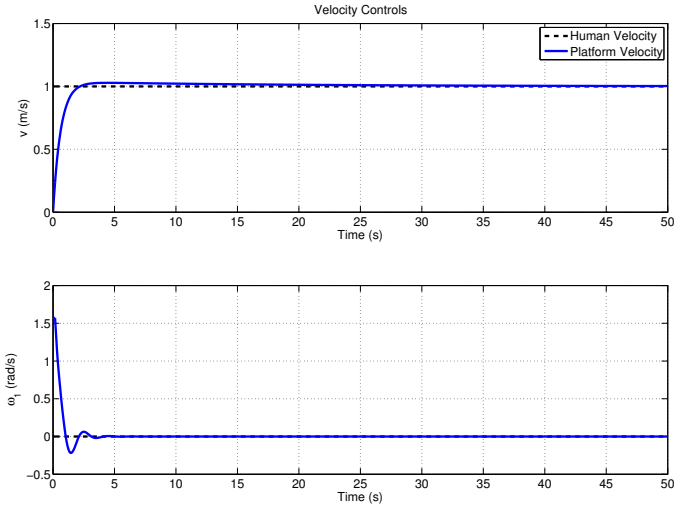
The proportional gain $K_4$ was selected so that the system bandwidth was approximately $1.75\,rad/s$. Once more, it is important to note that this results in a much faster response than the orientation regulating loop, because minimising crosstrack error is more important in the direction-based control scheme.

The controlled outer loop is closed loop stable, with poles at $z = 0.80 \pm 0.27\,\mathrm{i}$, corresponding to a damping factor of 0.468. This damping factor was selected so that the rotational velocity control does not exceed the platform maximum for the expected disturbances. A phase margin of 52.3° was obtained.

Figure 3.32 shows the disturbance response of the coupled system to a target offset by $t_x = 0.5\,m$, $t_z = 0.5\,m$ and $\alpha = 45°$, when the target moves forward in a straight line at $1\,m/s$.




(a) Plant response




(b) Controls

Figure 3.32: Disturbance response of system to a step change in an offset targets velocity when using direction-based control.

The direction-based controller causes all the outputs to be controlled, and results in a significantly faster reduction in cross track error $t_x$ and relative orientation $\phi$ as a result. Unfortunately, while this controller corrects orienta-

tion, it does so by traversing non-ideal trajectories. These trajectories make the system vulnerable to losing a fast moving target and are disconcerting as they often differ greatly from the path followed by the target. A more desirable trajectory is obtained through the traditional point-to-point controller approach.

Table 3.3 shows the gains used in the direction-based control scheme.

Table 3.3: Direction-based controller parameters.

| Parameter | Value |
|---|---|
| $K_3$ | 3 |
| $K_4$ | $-10$ |

### 3.6.5 Hybrid Control

Each of the previous controllers has both benefits and flaws. A new controller, combining the benefits of both these controllers could result in better system operation. A combined controller can be implemented through gain scheduling, or by phasing between the two controllers, depending on the target orientation.

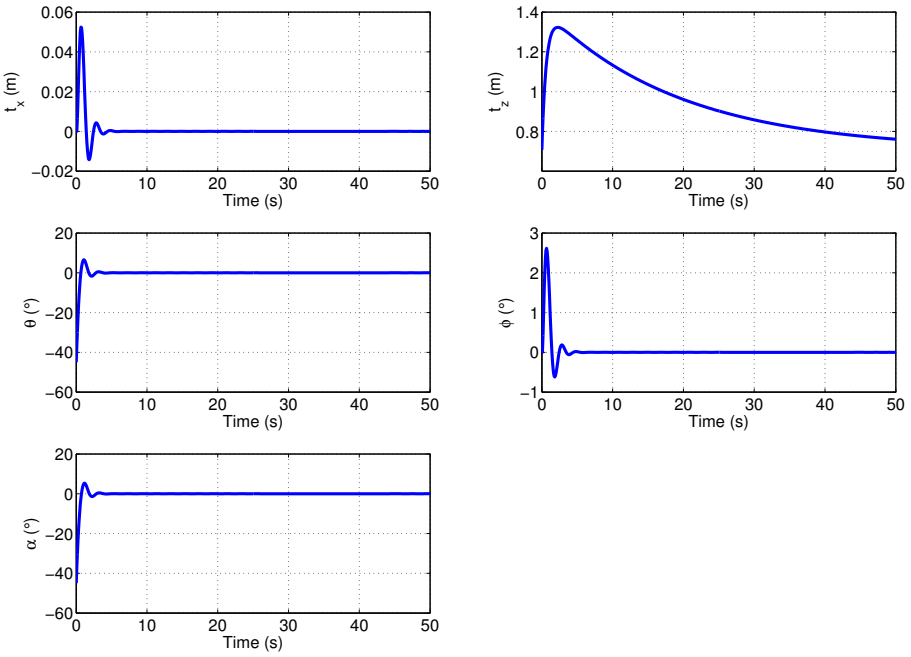
Using this phasing approach, the platform's combined angular velocity input at time step $k$ is generated by the weighted sum

$$\omega(k) = \left(\frac{|\phi(k)|}{\phi_{max}}\right)\omega_1(k) + \left(1 - \frac{|\phi(k)|}{\phi_{max}}\right)\omega_2(k), \tag{3.6.14}$$

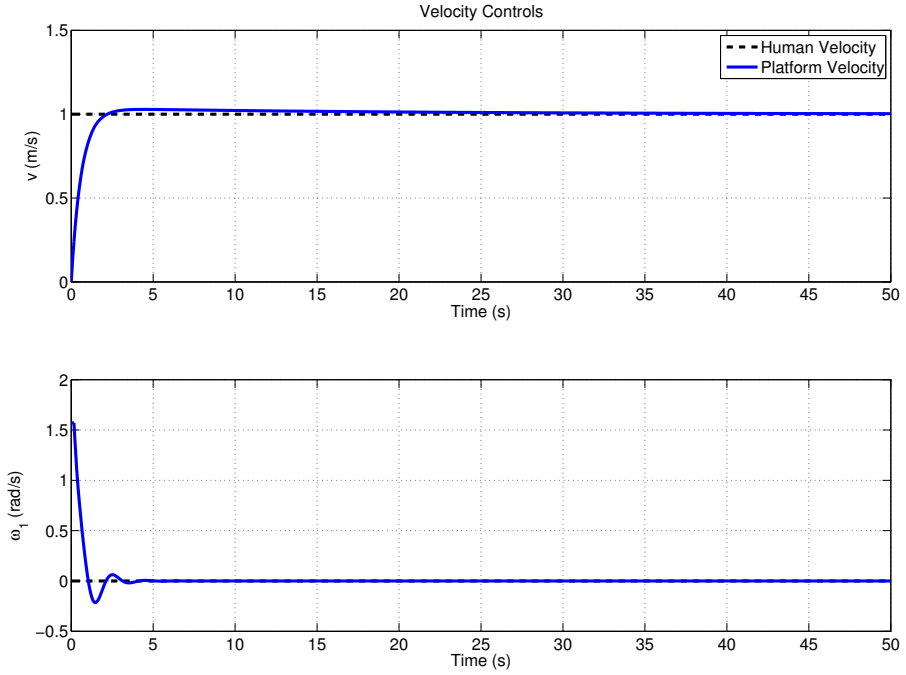
where $\phi_{max}$ is the maximum orientation angle detectable from an image. $\omega_1$ and $\omega_2$ are the rotational velocity control laws of Equations 3.6.9 and 3.6.11.

Once more, the forward velocity control law remains that of Equation 3.6.7. The operation of the hybrid controller is explained as follows. Should the target be turning sharply, emphasis is placed on the orientation-regulating controller so as to reduce the risk of losing the target during the turn. When the target orientation does not differ greatly from the platform's, the point-to-point positioner, which is less prone to losing a faster moving target, is preferred.

Figure 3.33 shows the disturbance response of the system to a target offset by $t_x = 0.5\,m$, $t_z = 0.5\,m$ and $\alpha = 45°$, when the target moves forward in a straight line at $1\,m/s$.




(a) Plant response




(b) Controls

Figure 3.33: Disturbance response of system to a step change in an offset targets velocity when using the hybrid controller.

The hybrid response looks similar to that of the direction based controller, but results in a slightly different trajectory. This trajectory, along with those of the point-to-point and direction-based controllers, is shown in Chapter 4, together with a more pertinent discussion on the benefits of each controller.

## 3.7 Summary of System Operation

A summary of the system operation and human-following approach is provided here for greater clarity. Figure 3.34 depicts a flowchart of the system operation and helps to show the order in which algorithms are applied.

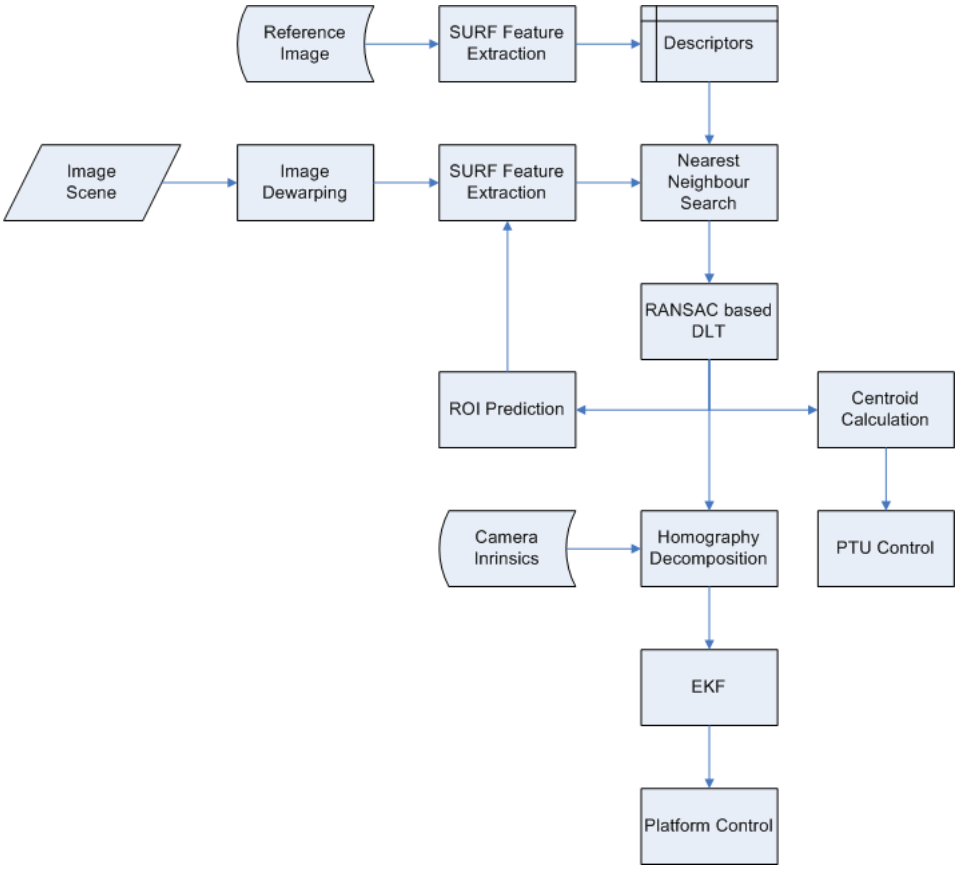



Figure 3.34: The flowchart illustrates the system operation and flow of information.

Initially, a template or reference image of the human target at the desired range from the camera is captured. The image of the human target is captured in a fronto-parallel configuration, so as to ease the extraction of pose. SURF features and descriptors are extracted and stored in memory for later use.

During system operation, an image scene is retrieved from a calibrated camera and dewarped. SURF features and descriptors are extracted, only considering a region of interest within the image scene. The extracted SURF

descriptors are compared with those of the template image by applying a naive nearest neighbour search. Confirmed feature matches are then used to extract a homography mapping features in the reference image to those located in the image scene, by applying the robust RANSAC-based DLT algorithm.

The extracted homography is used to predict the region of interest in which the target is next expected to appear, and to find the reprojected template centroid. The centroid is used to calculate the angle to target, which in turn controls the motion of the PTU.

The extracted homography is also decomposed into a rotation, translation measured up to scale, and a surface normal, by incorporating knowledge of the intrinsic camera matrix. Pose parameters of interest are then used by an extended Kalman filter to improve upon the pose estimates.

Finally, these pose estimates serve as inputs to a relevant control algorithm, which provides motion commands to the wheeled platform.

### 3.7.1 Implementation

Figure 3.35 shows the human-following robot used in this work. A commercially available Pioneer P3-AT mobile platform was used as a base for the integration of the software components presented here. This platform is controlled via serially transmitted actuation commands, using software provided by the manufacturer.

Figure 3.35: The mobile platform on which the work described in this thesis was implemented, together with a still image captured during a following task.

Software was designed in C++ using the Open Computer Vision software libraries and an open source robotic architecture (Candy *et al.* (2010)) developed by the Mobile Intelligent Autonomous Systems Group at the Council for Scientific and Industrial Research. The software architecture allows for easier message passing and modular node-based design using a publish/subscribe framework.

Experiments were conducted on a Dell Latitude 6400 dual-core notebook with 2 GB RAM. A Prosilica GigE Ethernet camera was mounted on a commercially available pan-tilt unit, that accepts pan and tilt commands through serial communications.

# Chapter 4

# Simulated and Experimental Results

The results of tests verifying and validating the performance of the selected problem solution are presented here. Initially, a summary of the object recognition results, presented earlier in Section 3.2.4, is provided, together with a discussion on the system limitations induced by the object recognition module. This is followed by simulations and analysis of the pose estimation modules, together with a discussion and motivation for the certainty measure introduced in Section 3.5.

Thereafter, the performance of the human pose estimator is discussed by analysing its performance over variations in angle and scale, followed by a description of both simulated and experimental results highlighting the performance and benefits of the three control systems presented in Section 3.6.

Finally, complete system results showcasing the performance of the human follower are provided, together with a discussion of overall system limitations. The improvements made through the use of the extended Kalman filter tracker are also highlighted here.

## 4.1 Object Recognition Limitations

The most important results of the feature-based object recognition system used in this work were discussed in Section 3.2.4. The following section serves to summarise these results and list the limitations that the object recognition system introduces. In addition, the influence of the hardware used in this work is discussed, together with the trade-offs required in the module design.

A Prosilica GigE Ethernet camera was used to test the object recognition modules. The camera imposes fundamental restrictions on the system operation. The first relates to the speed of image capture. Ideally, images should be captured as quickly as possible, with a very short exposure time so as to limit motion blur. This requires a large lens aperture, so as to increase the amount

of light entering the camera. If less light enters the camera, the exposure time needs to be increased, making the camera more susceptible to motion blur. The feature detection scheme will fail to operate if the image of the target is too blurred, as features will not be extracted.

Improving the robustness of the feature detection scheme motion blur induced by lighting is beyond the scope of this work, and it is assumed that lighting can be controlled relatively well. It is still important to recognise the role of the lens aperture, as the desire for a larger aperture and more light contrasts with other factors influencing system performance.

The second restriction on system operation results from the depth of field of the camera. A camera's depth of field refers to the range over which an object in an image remains sharp or in focus. The feature detection scheme will fail to operate if the image of the target is not sufficiently clear, as features cannot be extracted. The depth of field of a camera is typically increased by decreasing the aperture size, in contrast to motion blur, which is limited by increasing aperture size. This means that the depth of field will be limited by the allowable amount of motion blur in a fixed focus camera.

The final restriction of importance results from the camera's field of view. Ideally, the field of view should be as large as possible, to maximise the chances of detecting targets in a scene. The camera field of view is controlled by design of the camera lens. The field of view places a theoretical limit on the speed at which targets can be tracked, as targets will not be detected if they move across a scene in less time than the object recognition system can process an image.

An extremely large field of view is not always desirable, however, as much greater resolution images need to be processed to achieve the same feature detection and matching performance obtained using camera lenses with a smaller field of view, where targets occupy significantly more pixels. A larger resolution image results in greater processing time, even if the windowing and resolution control of Section 3.2.5 is incorporated.

It is clear that the performance of the object recognition system is heavily dependent on the camera and lens used. Great care needs to be taken to select camera and lens properties, given the trade-offs discussed above. In general however, improved object recognition performance can be achieved, through the use of improved cameras and lenses with more precise optics, although these are typically more expensive.

Table 4.1 provides a quantitative summary of the system limitations when detecting the All Bran box target. The optical axis range refers to the distance, in meters along the camera's optical axis, over which a target can be detected. Similarly, the target roll, pitch and yaw range refers to the allowable target orientations over which the object recognition system succeeds. These limits were measured independently, so the actual detectable range of any particular rotation may vary somewhat, given the remaining possible rotations. Note that the object recognition system is capable of achieving significantly greater

roll ranges (360° coverage), but this is constrained as a consistency check, since human targets are not expected to undergo roll motions greater than 90°.

Table 4.1: Qualitative list of object recognition system limitations.

| Property | Value |
|---|---|
| Optical axis range | $0.5\,m$ to $4\,m$ |
| Target roll range | $-90°$ to $90°$ |
| Target pitch range | $-20°$ to $20°$ |
| Target yaw range | $-60°$ to $60°$ |
| Processing frame rate | 11.58 fps |
| Field of view | $\pm 45°$ |

It is important to note that these parameters, measured experimentally, are dependent on the camera and lens used and vary depending on the experimental setup. The processing frame rate was measured by taking the system's average image processing frame rate on a 2.66 GHz dual core notebook with 4 GB RAM. The field of view is a parameter of the camera lens, but important to note as it introduces a constraint on the maximum speed of targets, measured perpendicularly to the camera's optical axis,

$$v = (2z \tan 45°)\,(11.58) = 23.16z \ m/s. \tag{4.1.1}$$

Here, $z$ is the target distance from the camera, along the optical axis. This corresponds to a theoretical worst case limit of $11.58\,m/s$ on the target motion, assuming a horizontally travelling target $0.5\,m$ away from the camera. While this is significantly faster than the average human's walking speed, it should be noted that the processing frame rate is not constant, and varies depending on the amount of clutter and number of keypoints detected in a scene. In addition, motion blur dramatically reduces the allowable target speed. More detail on the maximum target speed that can be followed is provided in Section 4.6.

## 4.2 Pose Estimation Simulations

Due to the difficulty in accurately controlling the attitude of a target, the performance of the pose estimation module is analysed through simulations incorporating most of the limitations and potential problems resulting from the object recognition module. The inclusion of the object recognition limitations allows for simulations closely matching reality, because the pose estimation process is not subject to external noise sources.

The simulations conducted consisted of 10000 trials, with each trial following the process described here. Initially, $N$ uniformly distributed points

on the plane $Z = 1\,m$ in 3D space are selected, with the $X$ and $Y$ positions constrained to between $-0.5\,m$ and $0.5\,m$. This ensures that points remain within 22.5° of the camera's 45° field of view.

Uniformly distributed roll, pitch and yaw angles are generated, with roll and pitch angles limited to between $-20°$ and $20°$. This corresponds to small rotations typically introduced by uneven terrain or unexpected target motions. Larger yaw angles, however, are expected and thus constrained to between $-60°$ and $60°$, corresponding to the object recognition system's detection limits for yaw motions (see Figure 3.19).

Similarly, uniformly distributed random translations are generated. Translation along the optical axis, $t_z$, is constrained to within $-0.5\,m$ and $3\,m$, which ensures that the target remains within the detection limits of the object recognition system, a range of $0.5\,m$ to $4\,m$. Horizontal and vertical translations, $t_x$ and $t_y$, are constrained to within $-t_z - 0.5\,m$ and $t_z + 0.5\,m$, which ensures that targets remain within the camera field of view when shifted, given the allowable positions of the 3D points generated earlier.

Thus far, no mention on the selection process for $N$, the number of 3D points generated, has been made. $N$ typically equals about 400 when the target is in a fronto-parallel position at a scale of 100%, but decreases with yaw rotations and scale changes (see Figures 3.19 and 3.20). For the purposes of the pose estimation simulations, this is modelled naively, under the assumption that the decrease in features due to scale changes and rotations is independent. Under this assumption, and approximating the distributions of Figures 3.19 and 3.20 with zero mean Gaussians, using standard deviations 15° and $1\,m$ respectively, the number of features is given by

$$N = 400e^{-\left(\frac{\alpha^2}{2(15)^2}\right)}e^{-\left(\frac{(t_z)^2}{2(1)^2}\right)}, \tag{4.2.1}$$

where $\alpha$ is the randomly generated yaw angle and $t_z$ the translation along the optical axis.

Figure 4.1 shows the resultant distribution used to select the number of features. Note that the choice of 15° standard deviation causes the yaw-based Gaussian to cut off at about 60°, the limit imposed by the object recognition system. Similarly, the $1\,m$ standard deviation results in a cut off of $3\,m$ for the scale Gaussian, corresponding to the range limitation of the object recognition system. The Gaussians ensure that the number of features generated and used for pose estimation at a specific target attitude is very close to that seen in practice, allowing for a realistic simulation.

Once the $N$ 3D points have been generated, they are projected onto a hypothetical image, using an intrinsic camera calibration matrix of

$$\mathbf{K} = \begin{bmatrix} 1800 & 0 & 512 \\ 0 & 1800 & 384 \\ 0 & 0 & 1 \end{bmatrix}, \tag{4.2.2}$$

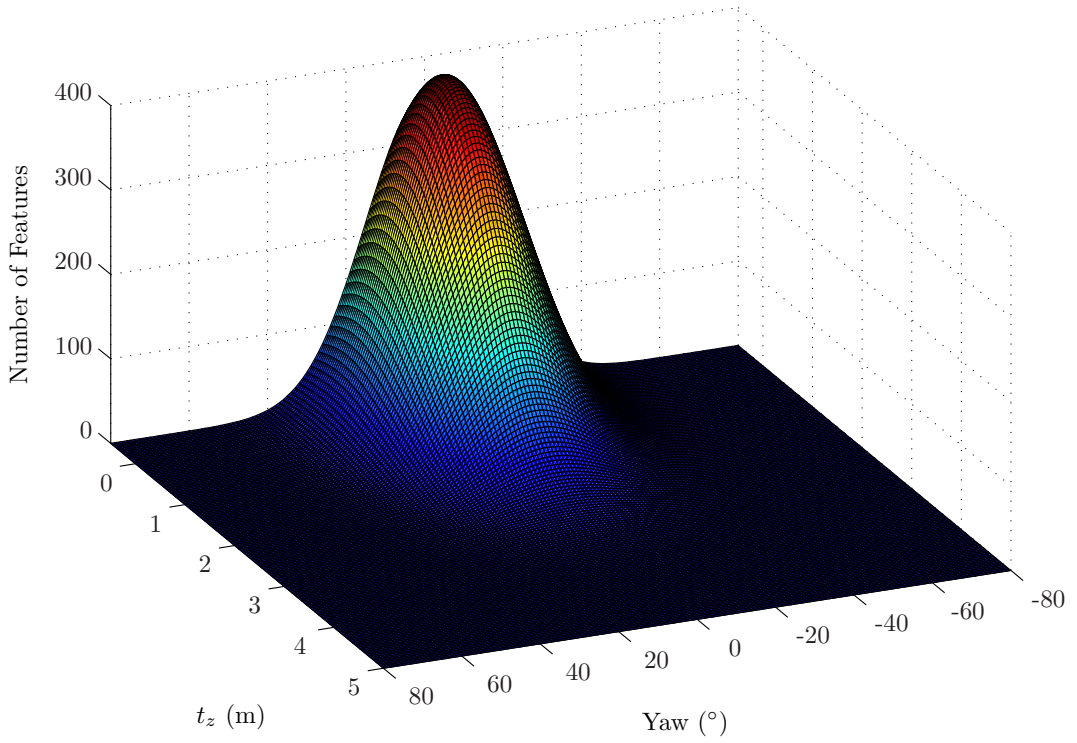



Figure 4.1: The distribution from which the number of features used in the pose estimation calculations is drawn. The number of features selected is dependent on the target attitude.

corresponding to an image of resolution $1024 \times 768$ pixels, and the camera lens focal length of approximately 1800 pixels. This produces the set of template features.

The input image features are generated by transforming the $N$ 3D points, using the aforementioned rotations and translations, and projecting the result onto the image plane, again using the intrinsic camera matrix $\mathbf{K}$. Zero mean Gaussian noise with variance 3 pixels is added to the $x$ and $y$ image plane coordinates of these features. This corresponds to a maximum error of approximately 10 pixels, the largest encountered by the object recognition system.

After generating the template and input image features, the RANSAC-based homography estimation of Section 3.4 is performed. The homography is then decomposed, using the process outlined in Section 3.3. The errors in roll, pitch and yaw angles are stored, together with those in the translations $t_x$, $t_y$ and $t_z$.

In addition, both the number of features used in the pose estimation and the certainty measure introduced in Section 3.5 are saved. The certainty measure is the ratio of inliers used for the homography computation to the total number of features that could possibly be detected in ideal circumstances (400).

Table 4.2 shows the mean and standard deviation of the errors in roll, pitch

and yaw rotations. The mean error remains below 2° for all rotations and the standard deviation is not much greater than that of the noise introduced in simulation. As expected, this shows that the angular estimates will typically be as accurate as the object recognition data it receives.

Table 4.2: Mean and standard deviation of rotation angle errors in simulated pose estimation.

| Angle | Mean (°) | Std (°) |
|---|---|---|
| Roll | 0.5902 | 1.7605 |
| Pitch | 1.2998 | 3.4906 |
| Yaw | 1.3584 | 3.5440 |

Table 4.3 shows the mean and standard deviation of the errors in the translations $t_x$, $t_y$ and $t_z$. As this data is simulated, with known 3D locations of feature points, the typically unknown scale in translation can be found and the actual and estimated translations can be compared directly. The mean error remains below $30\,mm$ for all translations with a standard deviation of less than $65\,mm$. This is low enough to allow for acceptable human following.

Table 4.3: Mean and standard deviation of translation errors in simulated pose estimation.

| Translation | Mean ($m$) | Std ($m$) |
|---|---|---|
| $t_x$ | 0.0166 | 0.0441 |
| $t_y$ | 0.0251 | 0.0648 |
| $t_z$ | 0.0195 | 0.0575 |

Figure 4.2 shows the distributions of the roll, pitch and yaw errors against the number of features used in the pose computation and the certainty measure of Section 3.5. Logically, the number of features represents a good measure of potential error, as more features should increase the chance of an accurate pose measurement. As expected, the figures show a negative correlation between the errors and these measures. The figure also shows that while the measures show similar correlations, the certainty measure of Section 3.5 is less susceptible to outliers than the number of features. This is to be expected, as the certainty measure introduced in this work does not take outliers into account.

It is important to note that these figures are error distribution plots, which show that the certainty measure is a quality indicator, but that good matches can occur with a low certainty measure. This makes sense, as a high variance in data does not necessarily imply an incorrect measurement. However, the shape of the distribution does provide an idea of worst case behaviour, and shows that the certainty measure can be of use in filtering.

Ideally, the relationship between errors and the certainty measure should be linear, for the purposes of Kalman filter tracking, but the defined certainty measure is still useful as it provides an indication of when a position estimate can be trusted. This is confirmed by the Pearson product-moment correlation coefficients[1] between the rotation angles and certainty measures, shown in Table 4.4. $\rho_{aN}$ denotes the correlation coefficient between the angle errors and the number of features used for the pose calculation, while $\rho_{aC}$ represents the correlation coefficient between the angle errors and the certainty measure.

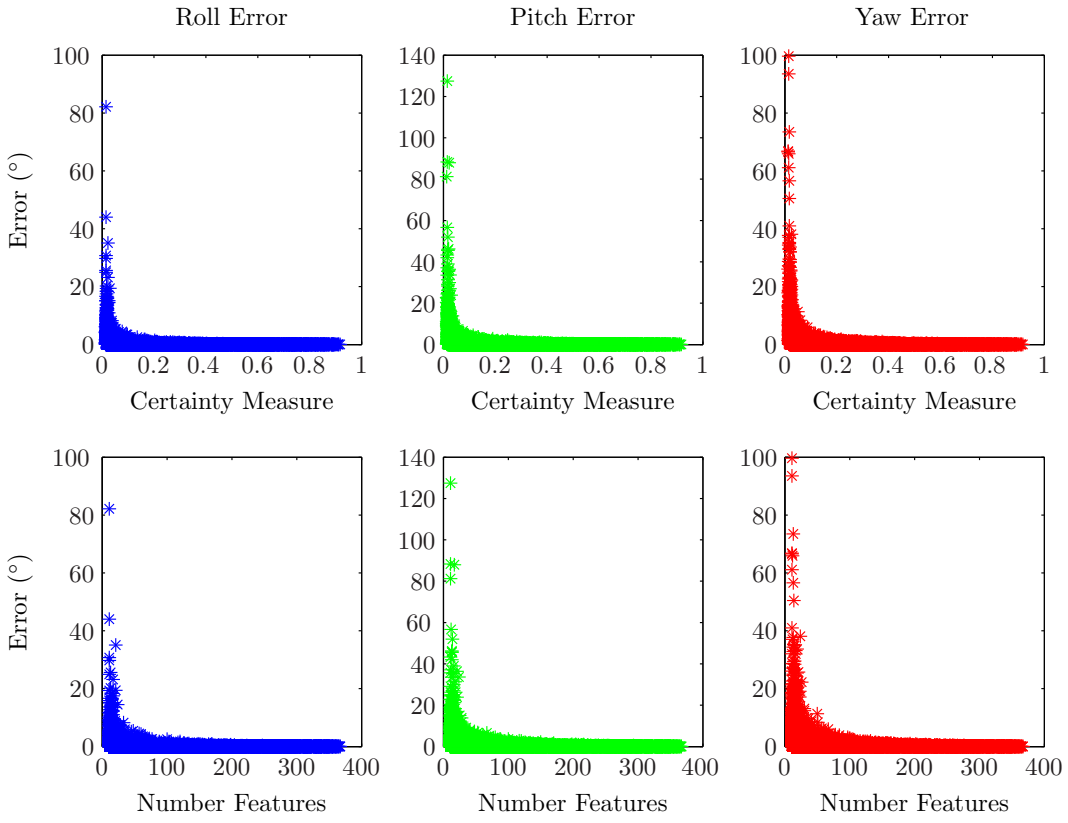



Figure 4.2: The relationships between roll, pitch and yaw rotation angles and both the number of detected features and the certainty measure defined in Section 3.5.

[1]The Pearson product-moment correlation coefficient, $\rho$, is a measure of the linear dependence between two random variables, $X$ and $Y$. $\rho = \frac{\mathrm{E}[(X-\mu_X)(Y-\mu_Y)]}{\sigma_X \sigma_Y}$, with $\mu$ representing the mean and $\sigma$ the standard deviation. E denotes the mean or expected value of the bracket's contents.

Table 4.4: Correlation coefficients showing the correlation between the errors in rotation angles and both the number of features used in the pose estimation calculation and the certainty measure defined in Section 3.5.

| Angle | $\rho_{aN}$ | $\rho_{aC}$ |
|---|---|---|
| Roll | −0.2281 | −0.2261 |
| Pitch | −0.2530 | −0.2527 |
| Yaw | −0.2627 | −0.2613 |

Figure 4.3 shows distributions of the translation errors against the number of features used in the pose computation and the certainty measure of Section 3.5. As expected, the figures show a similar negative correlation between the errors and these measures to that of Figure 4.2. Once more, the certainty measure of Section 3.5 is less susceptible to outliers than the number of features.

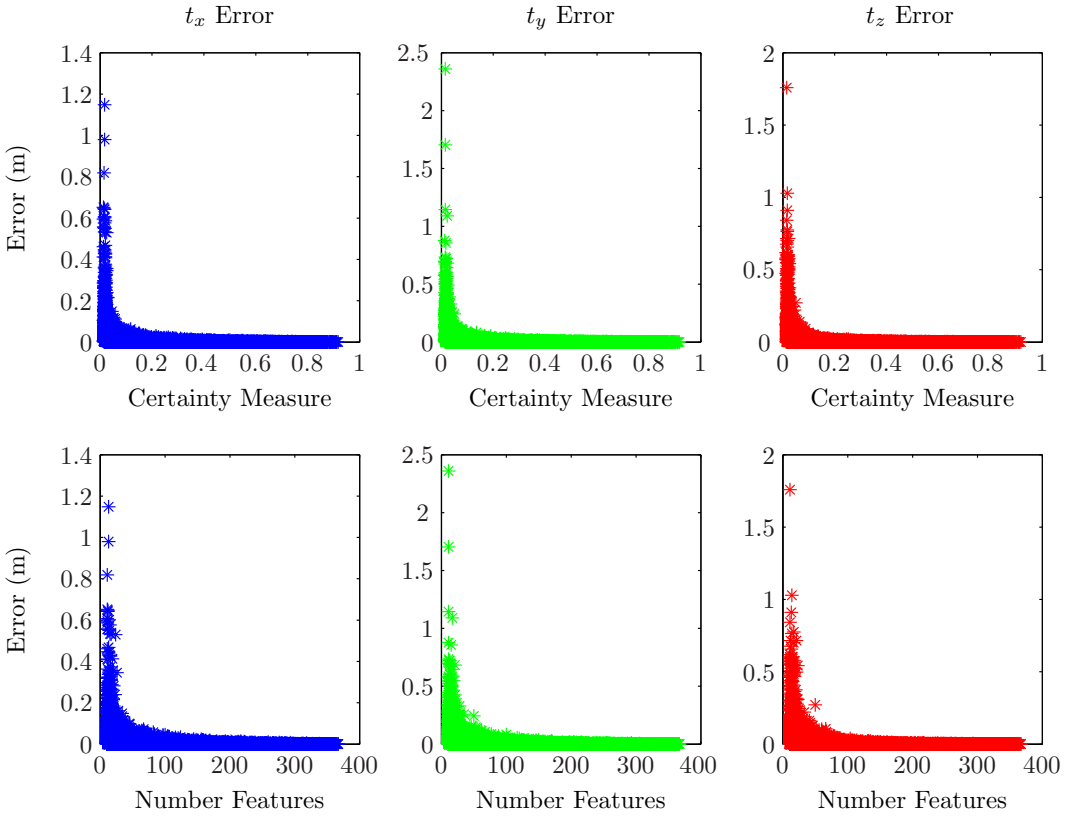



Figure 4.3: The relationships between $x, y$ and $z$ translations and both the number of detected features and certainty measure defined in Section 3.5.

As expected, the Pearson product-moment correlation coefficients between the translation errors and certainty measures, shown in Table 4.5, are similar

to those between the rotation errors and these measures and hence suitable for providing an idea of the accuracy in any given pose measurement.

Table 4.5: Correlation coefficients showing the correlation between the errors in translation and both the number of features used in the pose estimation calculation and the certainty measure defined in Section 3.5.

| Translation | $\rho_{tN}$ | $\rho_{tC}$ |
|---|---|---|
| $t_x$ | $-0.2539$ | $-0.2518$ |
| $t_y$ | $-0.2652$ | $-0.2642$ |
| $t_z$ | $-0.2474$ | $-0.2431$ |

$\rho_{tN}$ denotes the correlation coefficient between the translation errors and the number of features used for the pose calculation, while $\rho_{tC}$ represents the correlation coefficient between the translation errors and the certainty measure.

The results presented here have shown that the homography-based pose estimation process does not introduce additional errors, but typically performs as well as the input data supplied. More importantly, this section has shown that the certainty measure is a reliable indicator of the quality of a pose estimate.

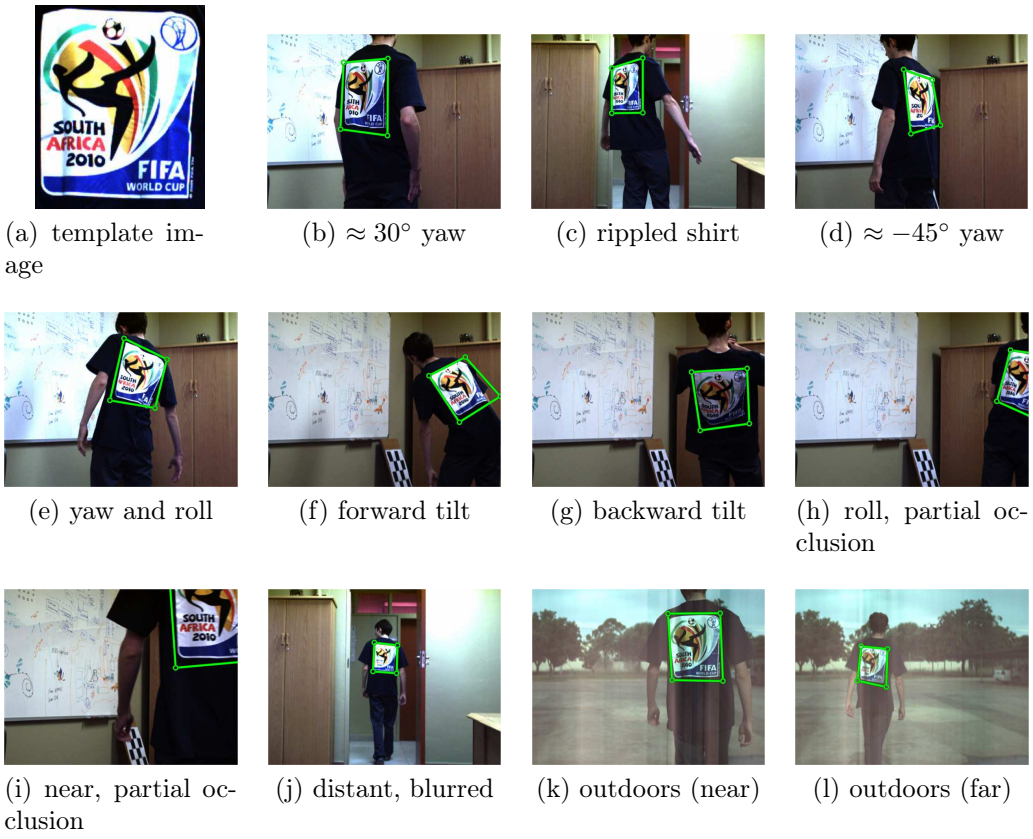
## 4.3 Human Pose Measurement Results

The following section aims to show that the homography-based pose estimate provides translations and a measure of orientation that is useful for the purposes of wheeled robot control. While we are unable to provide any insight into the accuracy of the measurement, as no ground truth is available, we aim to show that the homography-based plane fit provides a believable estimate of a human torso's facing direction.

Figure 4.4 confirms that the pose estimate is conceptually correct, through examples of planes fit through a human torso using the homography pose estimate. These examples show that a plane fit to the torso appears to capture a torso's facing direction. The reference image of the shirt worn during experiments is shown in Figure 4.4(a). All pose estimates obtained are measured relative to this view and the goal of a human-following task (using direction-based control) is to generate platform control signals that recreate this view.

The superimposed green quadrilateral in each example shows the estimated planar approximation to the back of the torso. As the figure shows, the system is robust when subjected to some extreme human motions and deforming clothing. Valid pose measurements are also obtained when the torso undergoes partial occlusions and over large scale changes. The images obtained outdoors are of poor quality and affected by glare, but illustrate that the system still functions effectively in challenging environmental conditions.

These images show that the pose estimate contains information regarding a person's position and orientation, but do not provide any information as to the accuracy of an estimate. As discussed in Section 3.3, only three pose parameters are of interest for the motion control of a wheeled platform: target yaw, the shift of the horizontal camera frame axis, $t_x$ and the optical axis shift, $t_z$. The results of experiments conducted to test the reliability of these measurements are now presented.




(a) template image (b) $\approx 30°$ yaw (c) rippled shirt (d) $\approx -45°$ yaw

(e) yaw and roll (f) forward tilt (g) backward tilt (h) roll, partial occlusion

(i) near, partial occlusion (j) distant, blurred (k) outdoors (near) (l) outdoors (far)

Figure 4.4: Results of the single-view homography-based pose measurement system on a range of test cases. The template image is shown in (a). The superimposed green quadrilaterals in (b)–(l) show the estimated planar approximations from which position and orientation, relative to the template, are extracted.

Figure 4.5 shows the relationship between actual variations of horizontal target motions and those obtained by the homography-based pose estimate. Image sequences of a stationary human target in a fronto-parallel configuration were captured at three positions approximately $2\,m$ from a camera. Measured values are linearly scaled and shifted so that there is a zero-mean, unity gain relationship between input and output, in order to facilitate comparisons.

Note that the homography-based pose estimate does not continually provide the same estimate when viewing a target in a static scene, as noise in

images causes changes in the features used for pose estimation. This variation in measurement is quantified by the standard deviation error bars displayed in the figure, with a line fit through the mean of estimates. The average certainty measures for each position are also noted in the figure.

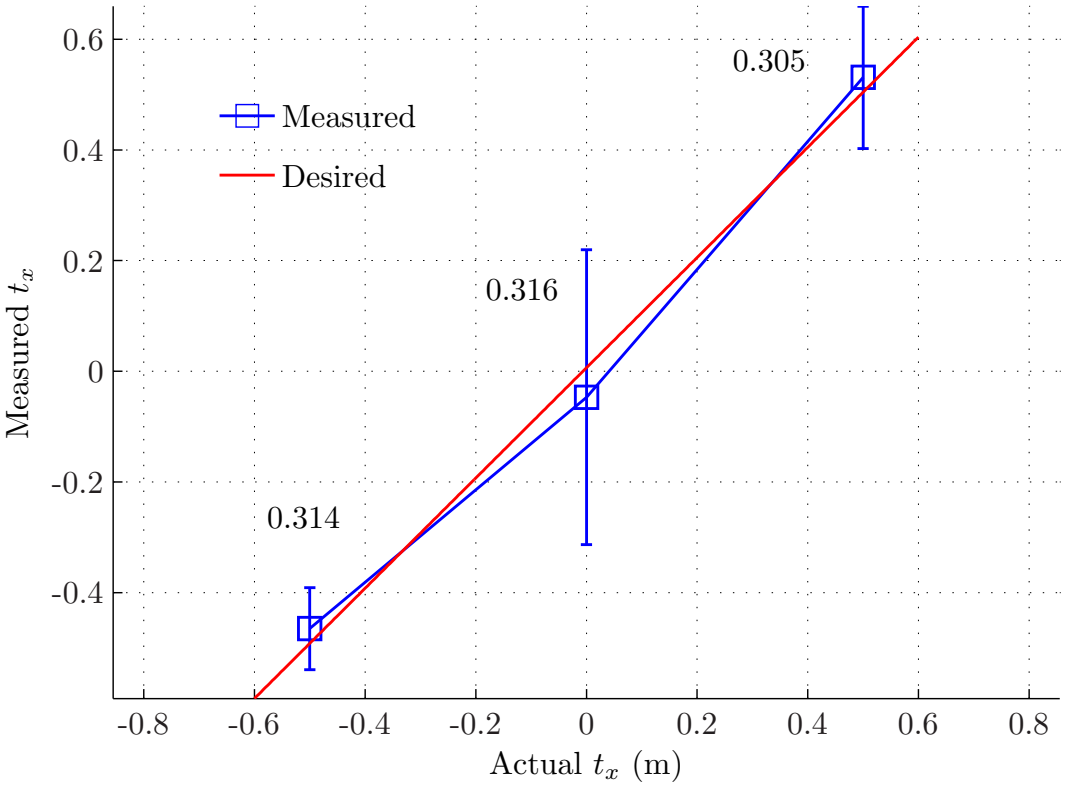



Figure 4.5: Mean (squares) and standard deviation (bars) of the measured horizontal translations given horizontal translation variations. Average certainty measures are also shown (the annotations alongside the error bars).

Figure 4.6 shows the relationship between actual variations of target motions along the camera optical axis and those obtained by the homography-based pose estimate. As before, image sequences of a stationary human target in a fronto-parallel configuration were captured at incrementing $0.5\,m$ intervals. The variation in measurement, quantified by the standard deviation error bars displayed in the figure, shows that the estimate becomes less reliable as the target moves away from the camera. The average certainty measures for each position confirm this. Once more, measurements were shifted and scaled to facilitate comparison.

Practical experimentation shows that the certainty measure rarely exceeds 0.6, with a measure greater than 0.1 corresponding to a reliable parameter estimate. Note that while the estimated translations are close to the actual translations used for test purposes, they are not exactly equivalent. This is unimportant for the purposes of controlling a platform using these parameters,

where suitable platform control can be obtained as long as the estimate is monotonic. This is clearly the case for the given translations, indicating that the estimates can be used for the purposes of control.

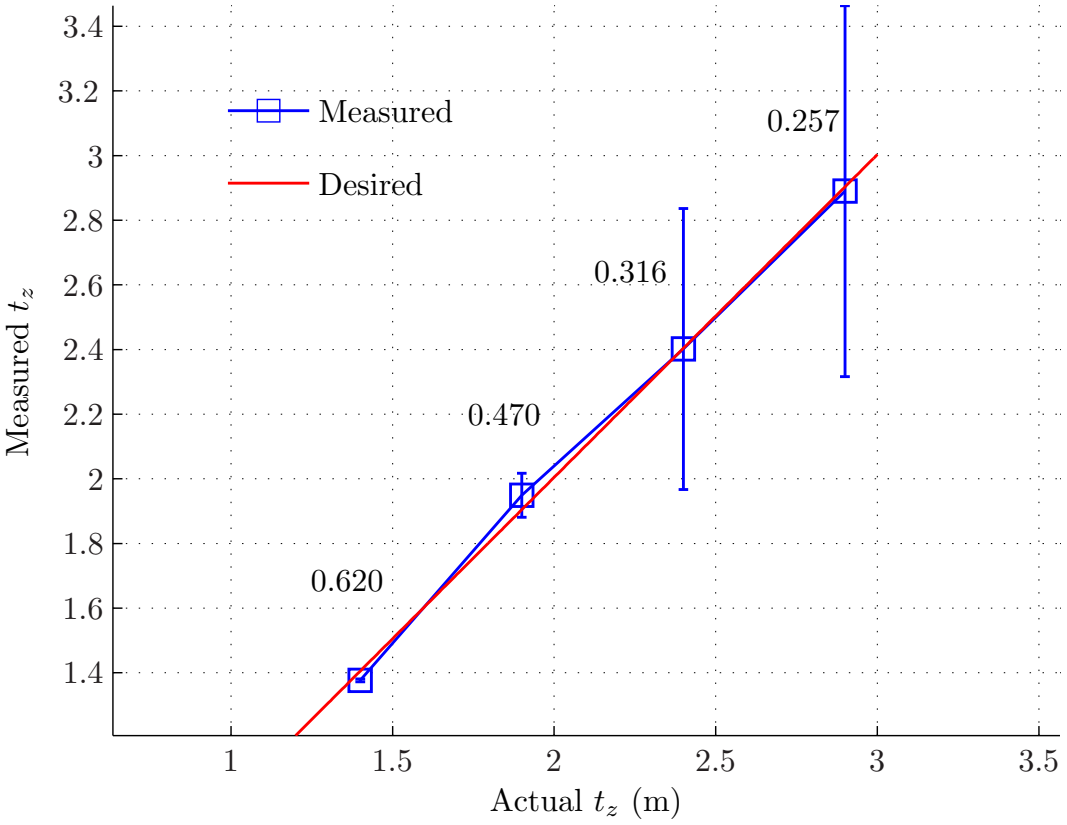



Figure 4.6: Mean and standard deviation, and average certainties, of measured optical axis translations against ground truth.

Figure 4.7 shows the relationship between actual variations of target yaw and those obtained by the homography-based pose estimate. These estimates were obtained by performing pose estimation on image sequences of a human target with varied orientation. Orientation was controlled by marking 20° intervals directly in front of a camera, and capturing image sequences of a human target facing in each of these directions. Once more measurements are shifted and scaled in order to ensure a unity gain, zero-mean relationship between measurement and input. The variation in measurement, quantified by the standard deviation error bars displayed in the figure, shows that the estimate becomes less reliable as the target rotates away from the camera. At ±60°, near the measured limits of the feature-based recognition, the estimate becomes untrustworthy. The average certainty measures confirm this.

The yaw measurements appear to be biased, detecting features at positive angles more accurately then those at negative angles. This is due to the distribution of features on the target, with more features detected on one side

of the target. A more symmetrical plot would be observed if the features were distributed uniformly across the target.

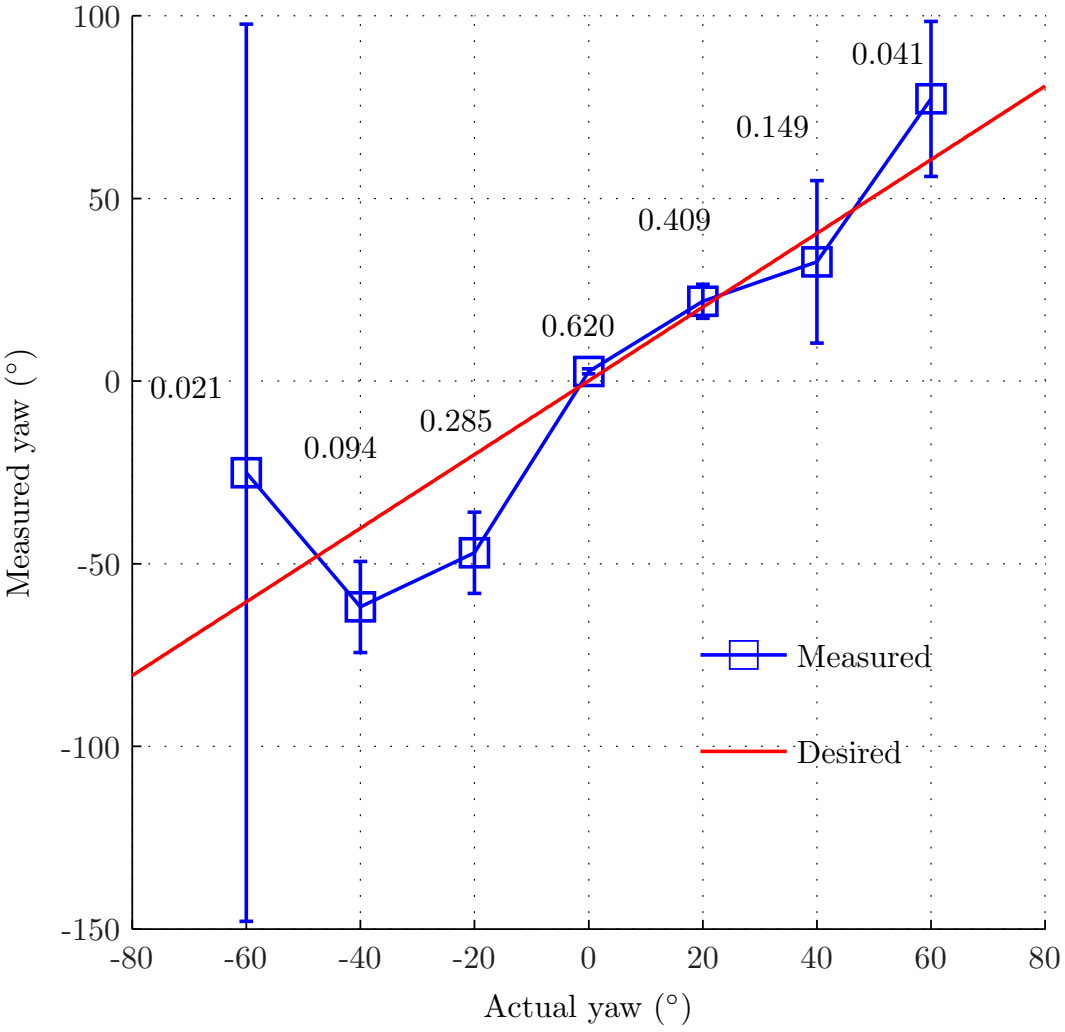



Figure 4.7: Mean and standard deviation, and average certainties, of measured target yaw against ground truth.

Once more, it is important to note that while the yaw estimates are not exactly that of the input system, this is unimportant for the purposes of control. As long as the measurements are monotonic, a suitable controller will still result in corrective motions that cause the magnitude of target yaw to decrease. This means that control should be successful as long as the relative target orientation falls between approximately $-50°$ and $60°$. In this region, corrective controller actions should result in target pose estimates that are more trustworthy, as indicated by the certainty measures in Figure 4.7, which would in turn allow for more accurate orientation control.

Figures 4.8 and 4.9 show the variation in target yaw, given pure translations. Ideally, the estimate should be independent, but in practice this is not achieved. Fortunately, this variation is not significant and does not affect the estimate's use in a control system.

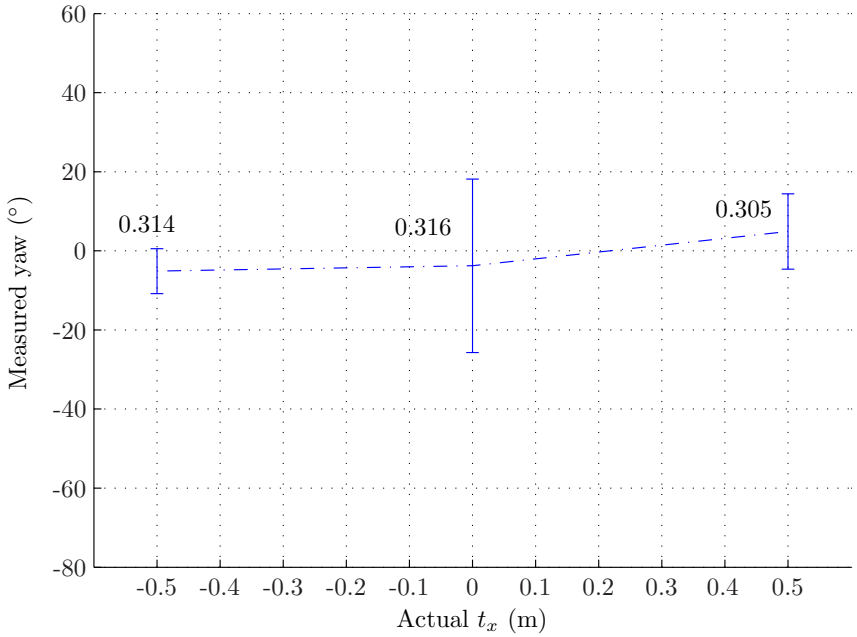



Figure 4.8: Effect of actual horizontal target translations on target yaw estimates. Average certainty measures are also shown.

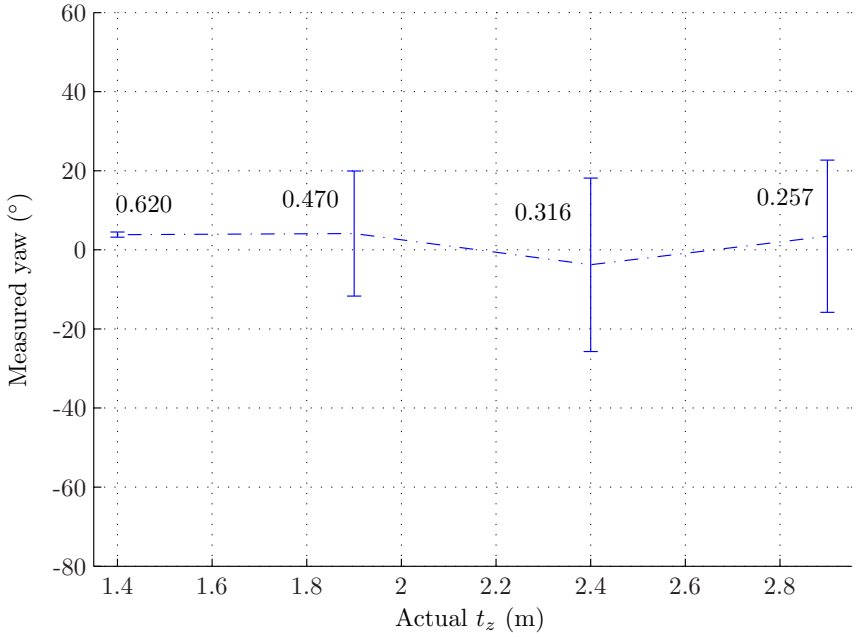



Figure 4.9: Effect of actual optical axis target translations on target yaw estimates.

Recall that the use of this estimate of human pose is still based on the assumption that a human's upper body represents a good measure of their travelling direction. The work of Anderson and Pandy (2001) reiterates this, with the finding that a human torso deviates within approximately 10° during a walking task. This finding implies that the non-ideal variation of target orientation given translations typically falls within the estimate noise floor, and is thus not overly significant.

## 4.4 Controller and Tracking Results

### 4.4.1 Controller Simulations

The intended behaviour of the three controllers discussed in this work is explained here, with the aid of simulated controller responses. As previously mentioned, the point-to-point controller causes the platform to move directly towards a target, with no control of the relative orientation between the platform and target. The direction-based controller causes the platform to move in such a way as to approach the target with the same orientation. In contrast, the hybrid controller attempts to combine benefits of both these controllers, by phasing between the two, depending on the relative target orientation angle.

#### 4.4.1.1 Controller Responses

Figure 4.10 shows the responses of the platform to a straight line trajectory, when each of these controllers are used. The target trajectory is offset from the platform starting position in both position and orientation.

The point-to-point controller does not converge to the path as quickly as the other controllers, since it only responds to the error in distance between the set-point trajectory and the platform. The direction-based controller, on the other hand, moves towards the straight line trajectory in an attempt to cancel out both relative orientation and positional errors. While this ensures that the direction-based controller takes target orientation into account, it results in a significantly longer distance travelled, and hence requires greater control action than the point-to-point controller.

The hybrid, gain-scheduling controller behaves as expected, producing a trajectory between the point-to-point and direction-based responses. This ensures that orientation errors are still reduced, but a shorter distance is travelled. Of course, this shorter distance comes at the expense of a slower reduction in relative orientation errors.

Figure 4.11 shows the response of the controllers to an offset circular trajectory. The response of the hybrid controller is not shown here, as its response is more-or-less identical to the direction-based controller in this case, since the relative orientation of a target following the circular trajectory is large enough to enforce a favouring of the direction-based controller in the hybrid scheme.

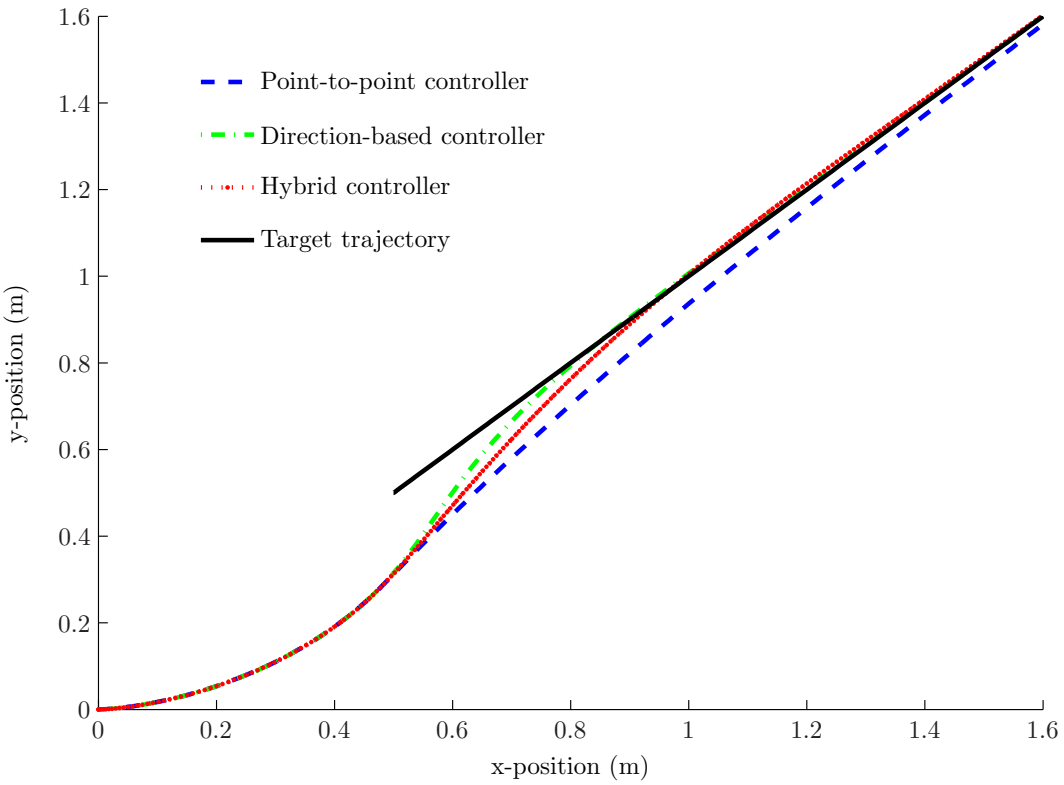



Figure 4.10: The simulated responses of the direction-based, point-to-point and hybrid controllers to an offset straight line trajectory.

The shorter distance travelled by the platform using the point-to-point controller is clearly visible here, along with the longer distance required by the direction-based controller. The direction-based controller essentially tries to control towards a tangent to the target trajectory, which thus results in a trajectory outside of the targets. This requires that the platform using the direction-based controller travel faster than the target, which in turn travels faster than the platform using the point-to-point controller.

The simulations provide valuable information regarding the expected limitations of each controller. The point-to-point controller, while traversing a shorter distance to the target, is vulnerable to losing sight of a sharply turning target. This could occur if orientation is not corrected and the relative target orientation moves beyond the orientation limit of the object recognition system.

While the direction-based controller corrects orientation errors and is not as susceptible to this problem, it does so by traversing a non-ideal path. This trajectory is longer, and requires greater actuation than that of the point-to-point controller. This potentially increases the chances of losing a fast moving target, which may leave the object recognition system's recognition range while the platform is attempting to correct orientation and positional errors simultaneously.

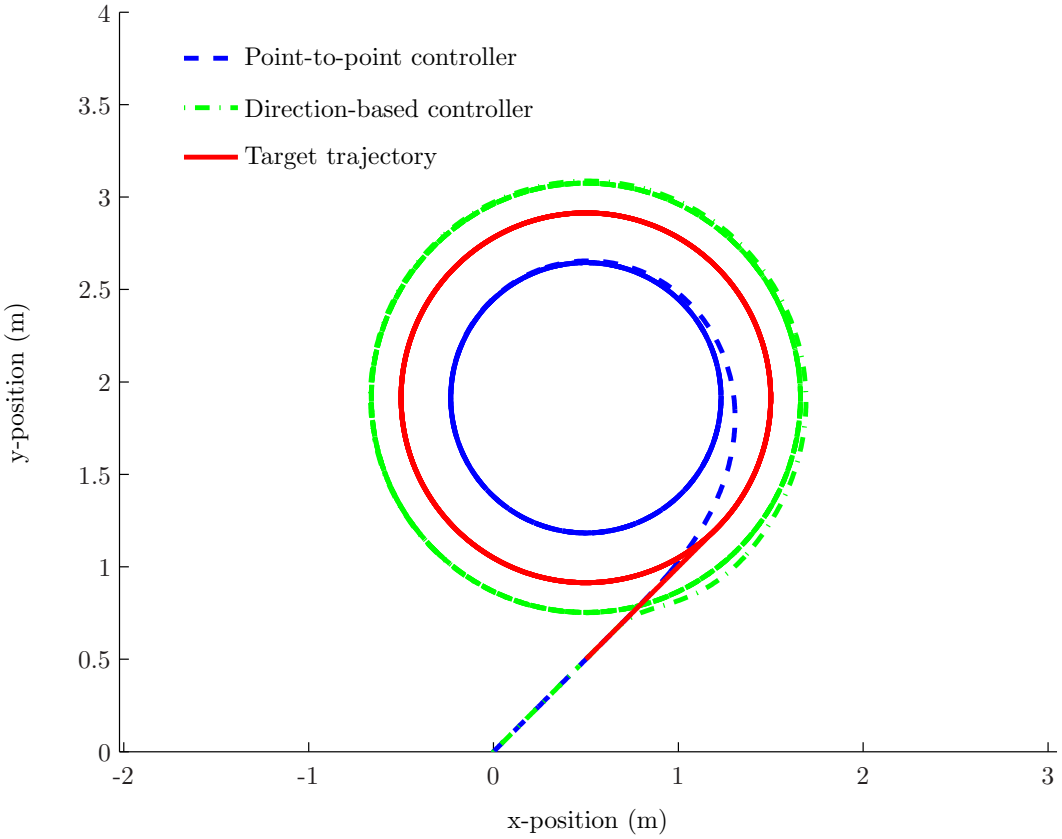



Figure 4.11: The simulated responses of both the direction-based and point-to-point controllers to an offset circular trajectory. The hybrid control response is not shown as it is almost exactly the same as the direction-based controller for this particular path. Note that the circular path was traversed more than once, causing the blue dotted and green dash-dot lines to appear continuous at later stages in the trajectory.

The hybrid controller, which combines aspects of the direction-based and point-to-point controllers, attempts to alleviate these potential problems, by producing trajectories that reduce orientation errors to some degree, but do not navigate along such non-ideal paths. Of course, in the attempt to combine the benefits of both controllers, a slight loss of performance may be experienced in the areas in which each controller performs strongly.

#### 4.4.1.2 Monte Carlo Analysis

The stability of the control systems described here is difficult to analyse, as no closed-form model of the vision components is available. As a result, Monte Carlo analysis is used in an attempt to show the expected system bounds.

As our human-following system is already constrained by its allowable motion, we only consider forward and rotational velocities less than or equal to the maximum platform velocities. It is clear that any target motions exceeding these velocities cannot be followed.

Moreover, our analysis is only concerned with the platform velocities that can be followed, and assumes the platform starts in the desired position relative to the target, with no offset in position or orientation.

The simulations consisted of 1000 runs, each conducted as follows. Initially, uniformly distributed random forward and rotational velocities were generated. These velocities are used as controls for the target and remain constant for each iteration of a simulation run. The paths traversed by a platform when following a human target are not important here, but rather whether the platform is able to maintain sight of the target over each simulation. If this occurs, the following task is considered successful.

Initially, the target position and orientation is adjusted using the unicycle motion model in (2.5.1). The relative yaw, translations and pan angles used in the controllers presented earlier are then calculated. These parameters form inputs to the model developed in Section 4.2 and given by (4.2.1). This model approximates the number of features detected, given relative target position and orientation and is based on practical system measurements. A slight change is made to the pixel noise added in the simulations, which is increased from a standard deviation of 3 pixels to 10 pixels, in order to account for worst case noise introduced by potentially deforming clothing.

Recall that a certain number of features is required if the object is to be recognised successfully, so the number of features generated provides a termination criterion to the simulation. If sufficient features are generated, the relative target position and orientation are generated following the same approach as Section 4.2.

These parameters are then used as inputs to the relevant control system, which generates platform velocities. The platform velocities are used to update the platform motion, using the unicycle motion model in (2.5.1). A delay corresponding to the average image processing rate is also incorporated here.

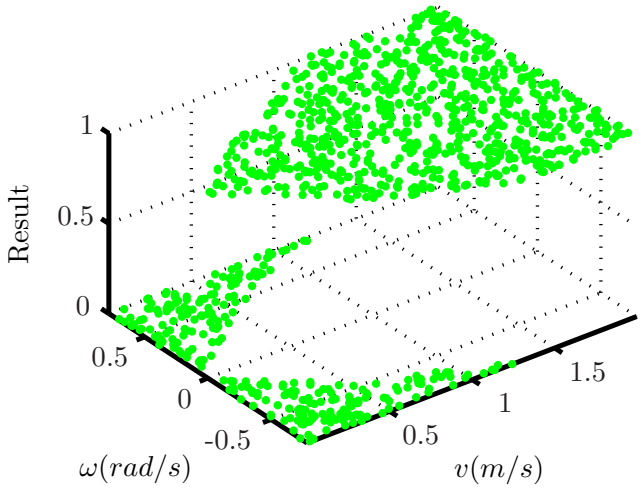
This process continues until a target is lost or a specified distance has been travelled. If the specified distance is travelled and the target was not lost, the following task is assumed to be successful.

Figure 4.12 shows results of the simulations for each of the three controllers proposed here. The figure confirms that while the point-to-point controller is able to follow rapidly moving targets, it is unable to follow sharply turning objects with little forward velocity. The direction-based controller is less sus-

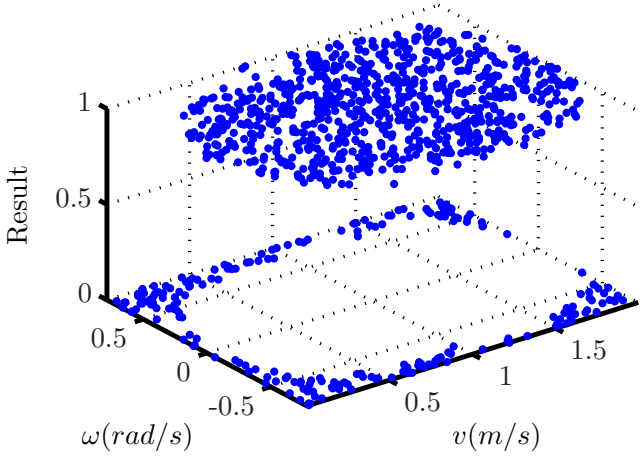
ceptible to losing targets that are sharply turning, but experiences difficulties when following rapid targets. The hybrid controller dramatically improves the following performance of both controllers, only experiencing difficulties in following sharply turning targets. This indicates that the hybrid controller should be better equipped to deal with human motion.

It is important to note that the stability of the controller has not been confirmed here and that the simulations have only covered constant velocity motion within the bounds of the allowable platform velocities. Other motions could still cause the platform to lose sight of a target. In addition, the simulation is model based and only an approximation of the physical human-following task.

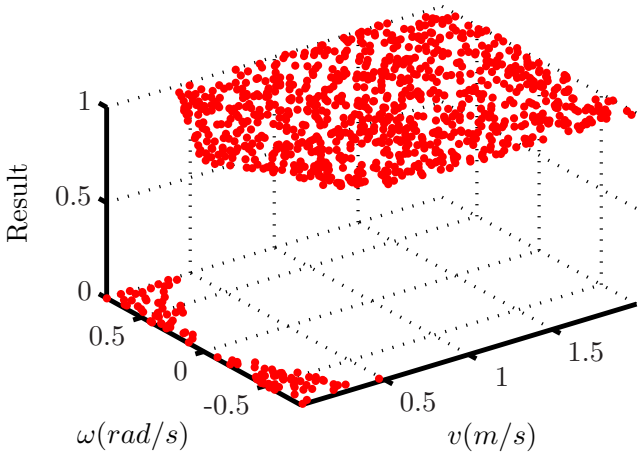
However, the simulation does provide good evidence as to the operation of the three controllers, and confirms our proposition that the hybrid controller offers better performance than its components.




(a) Point-to-point controller




(b) Direction-based controller




(c) Hybrid controller

Figure 4.12: Results of the Monte Carlo target-following analysis are shown here. A successful target-following task is denoted by a 1 on the result axis, while a failure is denoted by a 0.

### 4.4.2 Actual Controller Results

Actual results of the controller responses to target trajectories are now presented. Unfortunately, it is difficult to obtain ground truth when following a human target, so the responses presented here were generated by following a second robotic platform with a salient planar target attached. While the pose measured using a planar target is not as noisy as that obtained from a human torso measurement, it still represents a good approximation and allows for the actual behaviour of controllers to be examined.

The responses presented here were measured using odometry obtained from the platforms. While this odometry is subject to drift, it is reliable over short distances and hence a sufficiently accurate measure of position for our purposes. Note that while platforms are equipped with odometry, the system does not make use of this information and is purely vision based.

Figure 4.13 shows the step response of the three controllers to a target offset in both position and orientation.

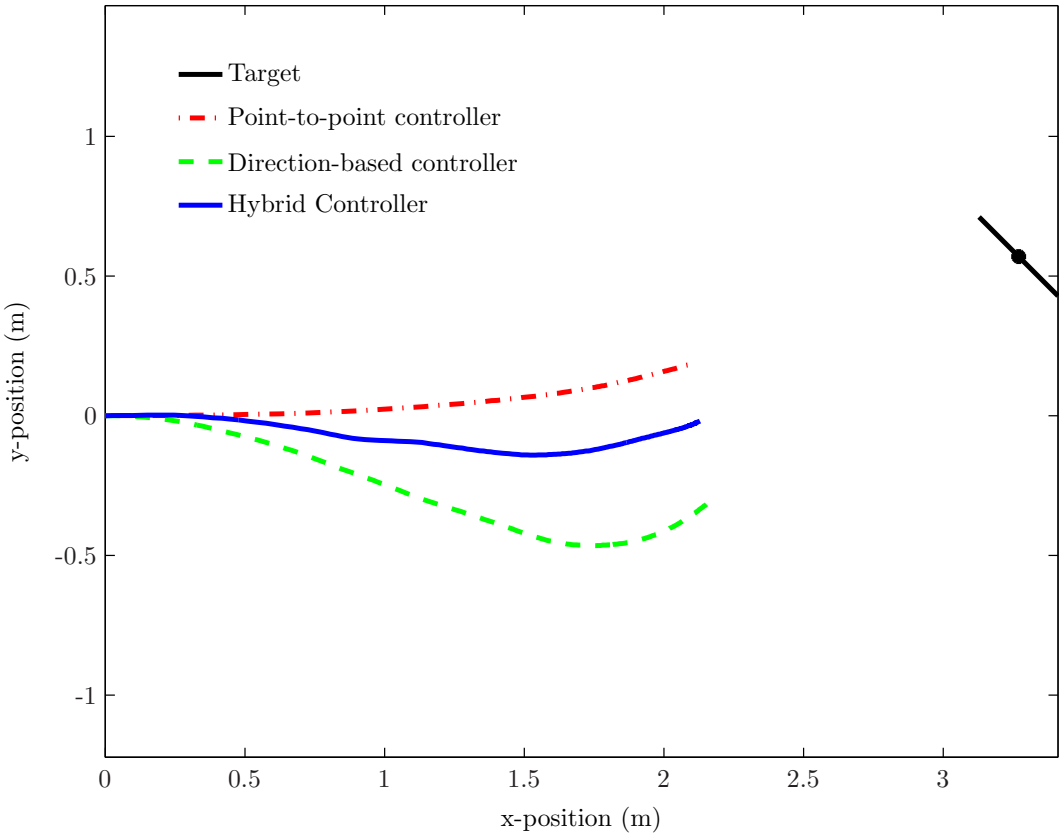


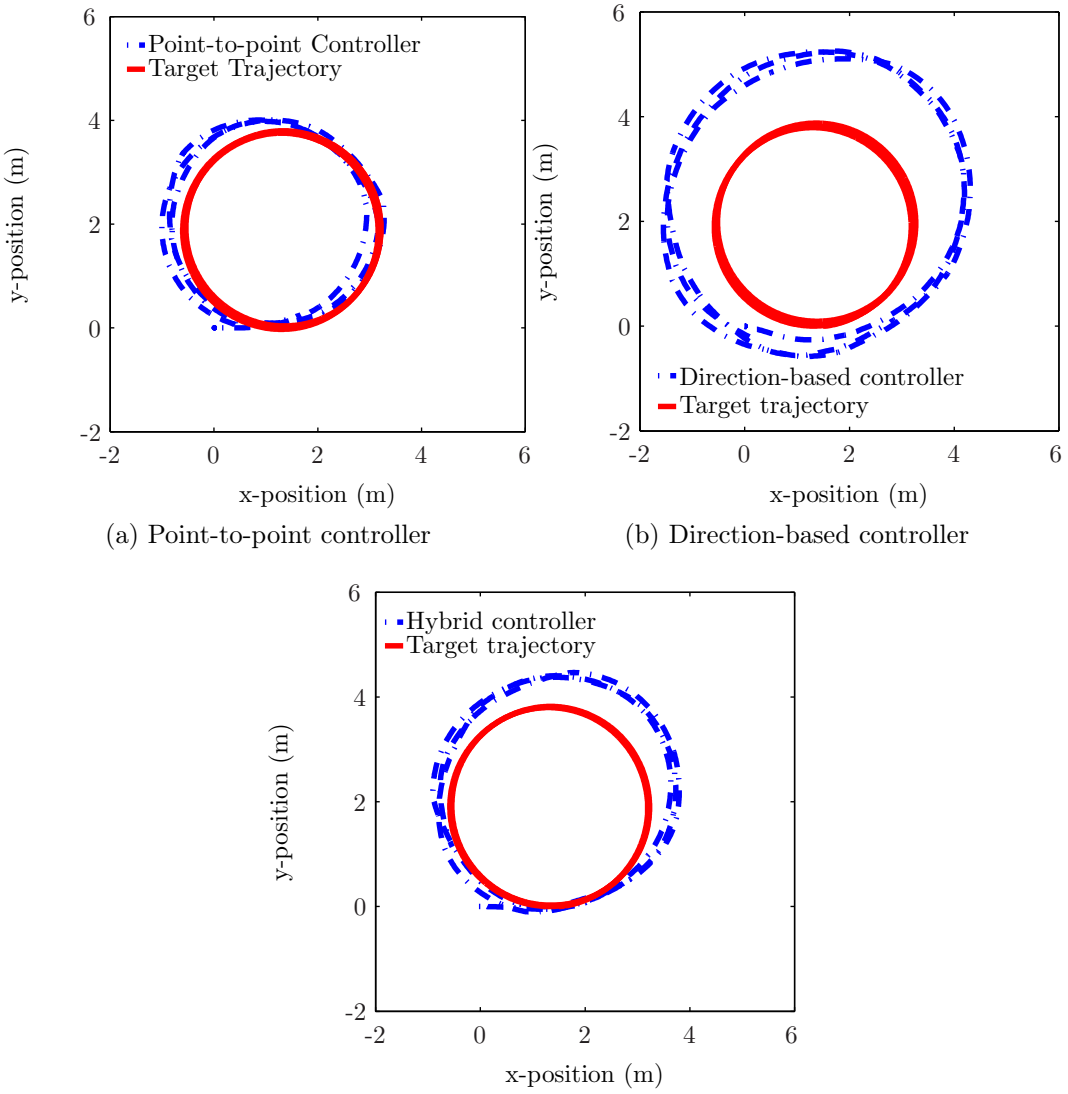

Figure 4.13: The responses of the three controllers to a stationary target object offset in both position and orientation.

The figure shows that the controller behaviour is close to that exhibited in simulation. The direction-based controller corrects relative orientation er-

rors, with an end relative orientation error of 5°. The point-to-point controller ignores orientation errors, which results in an end orientation error of approximately 30°. The hybrid controller corrects errors to an extent, but not completely, ending with an orientation error of approximately 20°.

While this difference in behaviour does not seem significant, it is important, as a pose estimate obtained in the 20° range has much greater certainty than one obtained at 30°.

Responses of the controllers to circular trajectories are shown in Figure 4.14. The responses are as expected and confirm the controller behaviour exhibited in simulation.




(a) Point-to-point controller

(b) Direction-based controller

(c) Hybrid controller

Figure 4.14: The responses of the three controllers to a target object following a circular path.

## 4.5 Human-following Results

In this experiment a human walked along a predetermined path with the robot following from a preset starting position. Internal robot odometry measurements were logged and are displayed for comparison with the predetermined path in Figure 4.15. Although odometry is subject to drift, and the starting point and orientation of the platform could not be accurately controlled, a good idea of the various controller behaviours to the same following task is obtained. Ground truth was obtained by manually marking out a path to be followed by a person and navigating the mobile platform along this path by hand, while logging odometry information.

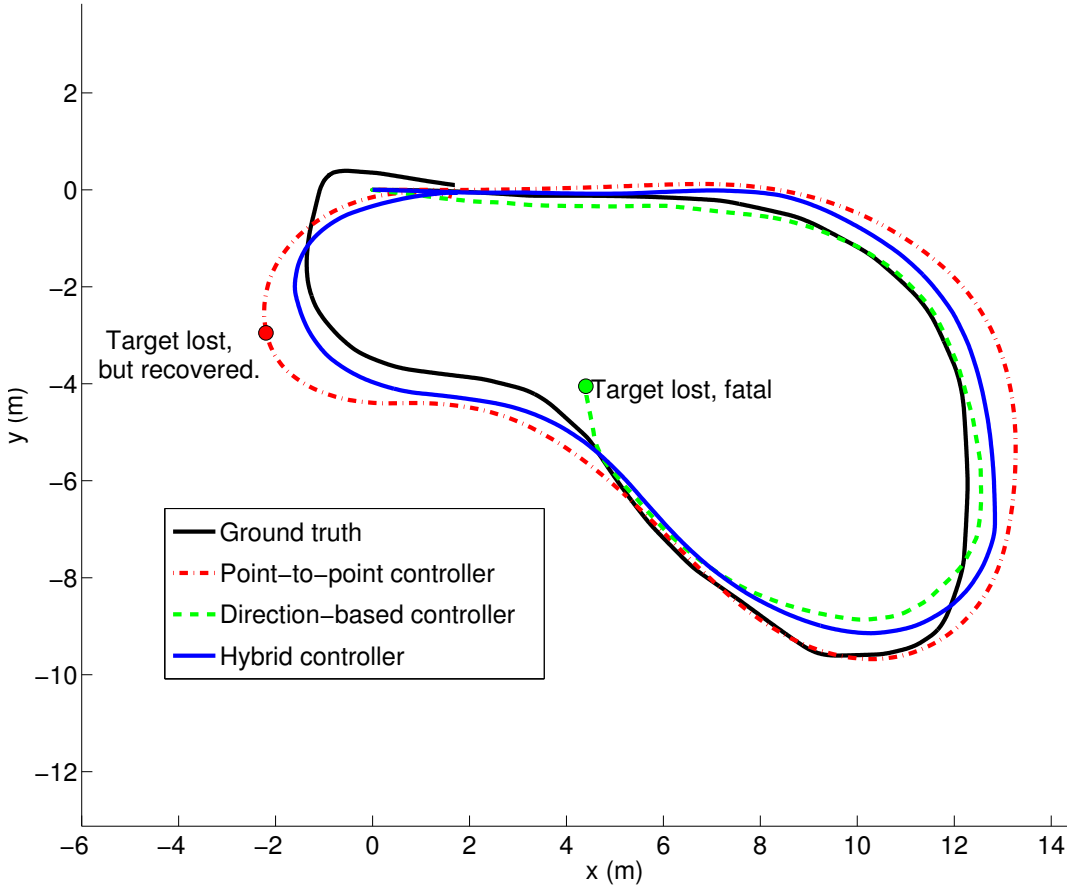



Figure 4.15: Position responses of the three controllers to a human target following a predetermined path. Both the point-to-point and direction-based controllers failed, the latter fatally so, while the hybrid scheme managed to follow without failure.

As indicated in the figure, both sub-controllers failed during the experiment. However, the failure of the point-to-point controller was not fatal, as (by sheer luck) the marked trajectory brought the target back into view and the platform was able to continue. As expected, the hybrid controller was successful, suitably overcoming the conditions that caused the sub-controllers to fail.

Figure 4.16 offers a better indication of the behaviour of the hybrid controller, as it shows ground truth target positions of both human and robot captured simultaneously. This information was obtained by capturing a following task using a stationary, forward facing laser scanner and using a laser-based target tracker (Burke (2010)). The tracking system allows for accurate trajectories of targets to be captured by separating returns of a forward facing laser scan into those of individual targets, and modelling the motion of these objects. The laser-based target tracker uses a four bin descriptor consisting of principal components and target position for matching, together with Kalman filtering to track targets in a scan.

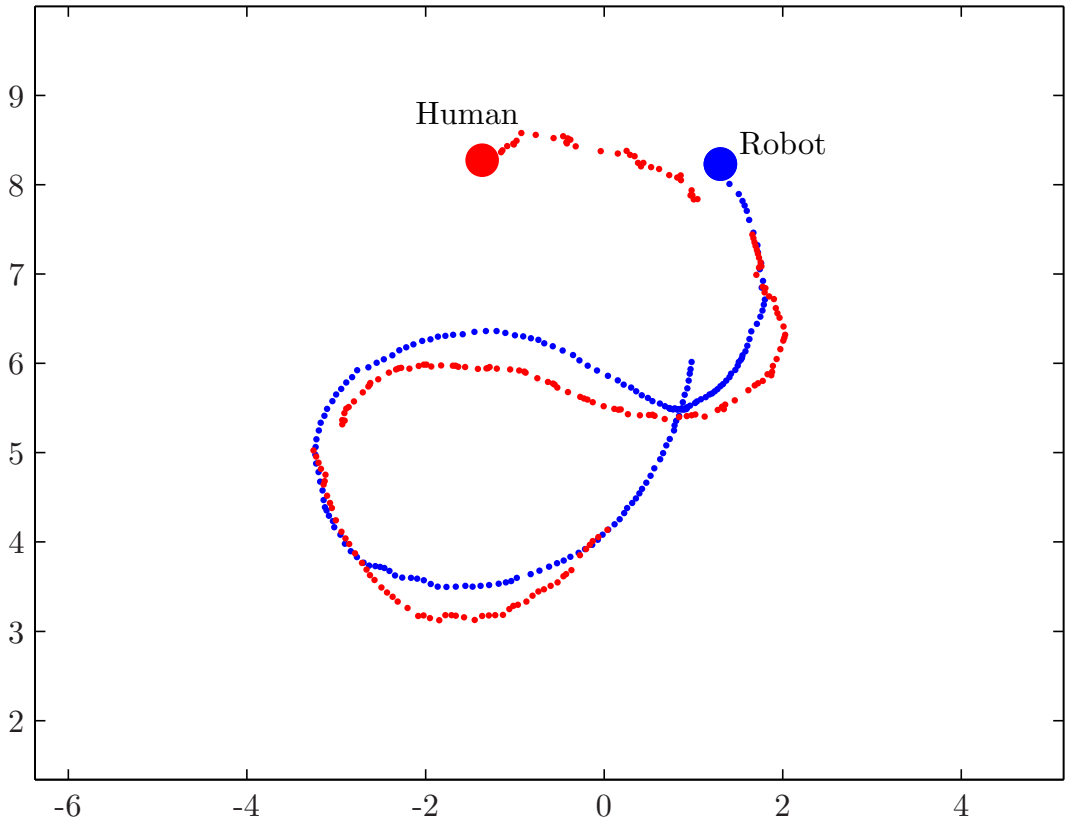



Figure 4.16: The paths followed by a human leader and following robot when using the hybrid controller. The path followed by the robot is marked in blue, while that of the human is in red. The end positions of the platform and human are marked as coloured circles. Portions of the human trajectory are missing in areas where the robot occluded the human in the laser scan.

The figure shows that the hybrid controller causes the robot to choose trajectories that aid in reducing orientation errors, while still following the human successfully. This is especially clear at the end of the path displayed, where the platform is delaying a turn in order to reduce relative orientation errors.

Note that only a brief, continuous section of the path followed is shown here for the purposes of clarity, but the system is able to follow a target walking naturally at slow to medium speeds for extended distances and time periods.

## 4.6 Motion Blur Limitations

The simulated results presented thus far have indicated that the gain-scheduling controller is able to follow targets at speeds of up to $2\,m/s$. Unfortunately, in practice, the allowable target speed is significantly lower, due to the effects of motion blur.

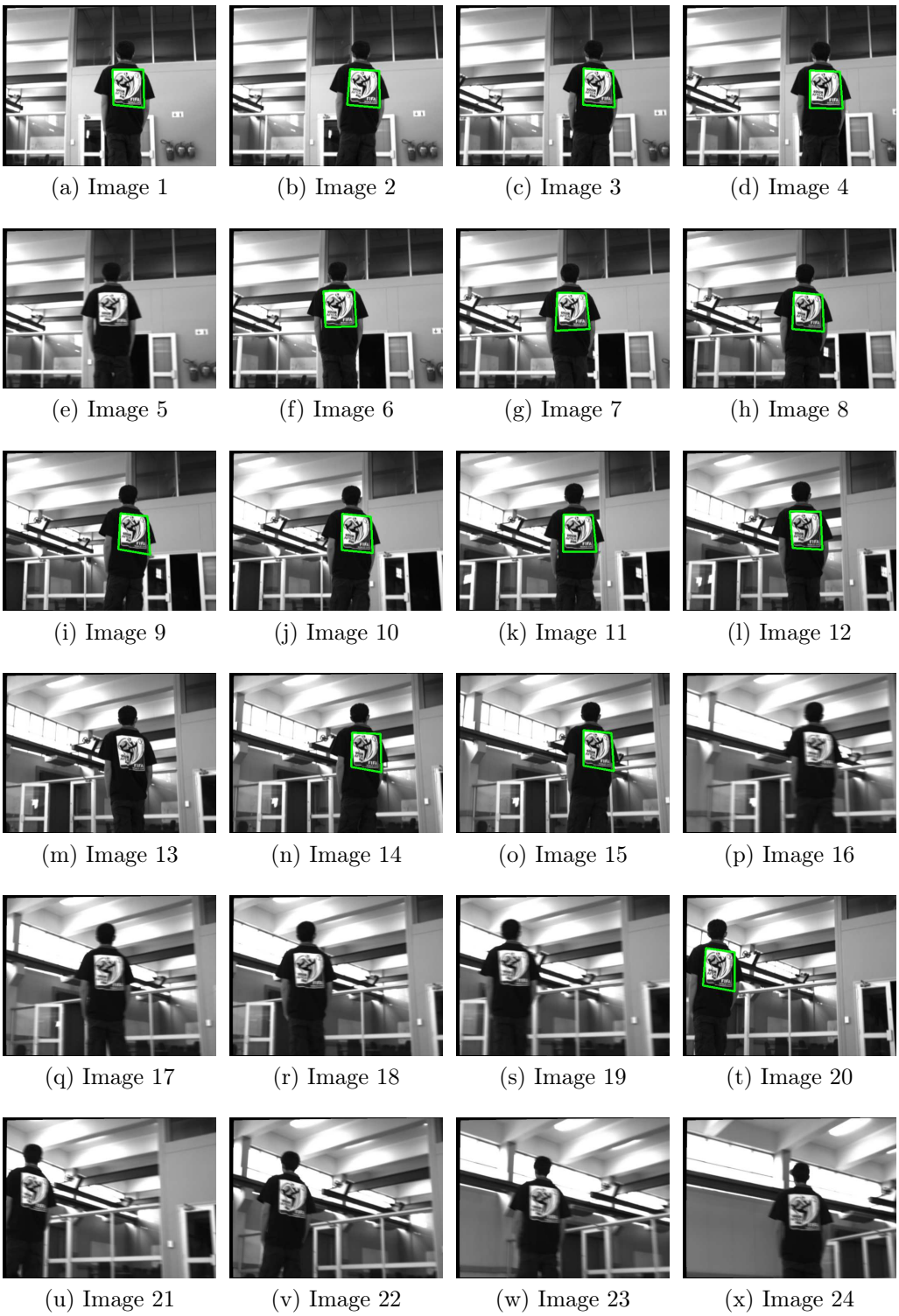
Practical experimentation, using the aforementioned laser tracker to determine human walking speeds, shows that the system is capable of following a target at approximately $0.7\,m/s$. This is equivalent to a slow to medium walking speed.

The simulated results indicate a much greater target speed, as they do not consider the effects of motion blur. As discussed in Section 2.2.1, features selected by feature-based matching schemes are usually corners, edges or salient points that differ from their neighbours. Motion blur introduces significant smoothing or removal of such information. As a result, insufficient features are extracted in the presence of motion blur.

This causes feature-based recognition and pose estimation strategies to fail. While predictive tracking such as the Kalman filter approach described in Section 3.5 does assist in countering motion blur, it is only able to do so for a brief period of time, after which a following task will fail in the presence of continued motion blur.

Figure 4.17 shows a sequence of images captured during a following task with challenging target motions. The sequence shows how measurements fail in the presence of motion blur. Figure 4.18 shows the system measurements and controls over this image sequence. The figures confirm that the Kalman filter tracking is effective at compensating for brief target losses due to motion blur, but that losses for an extended time period cause problems. This behaviour is exhibited most clearly by the velocity controls of Figure 4.18.

The system ramps down all velocities to zero when a target is lost. The momentary dip in the velocity controls of Figure 4.18 is due to this behaviour. While the platform is able to recover from this motion in some cases, it can potentially destabilise the system.



(a) Image 1 (b) Image 2 (c) Image 3 (d) Image 4

(e) Image 5 (f) Image 6 (g) Image 7 (h) Image 8

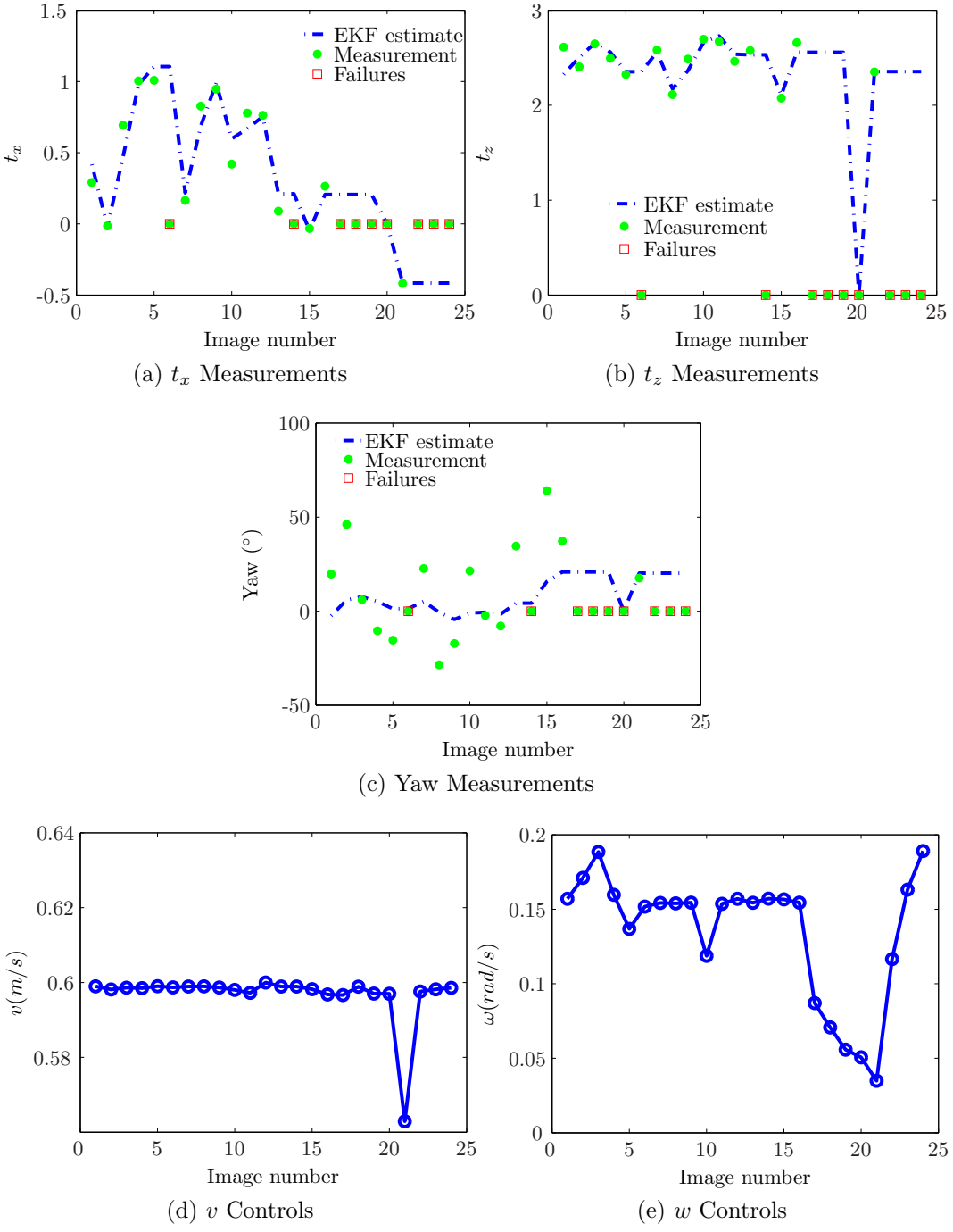
(i) Image 9 (j) Image 10 (k) Image 11 (l) Image 12

(m) Image 13 (n) Image 14 (o) Image 15 (p) Image 16

(q) Image 17 (r) Image 18 (s) Image 19 (t) Image 20

(u) Image 21 (v) Image 22 (w) Image 23 (x) Image 24

Figure 4.17: An image sequence of a target, captured during a human-following task. The target is lost in the presence of motion blur.




(a) $t_x$ Measurements

(b) $t_z$ Measurements

(c) Yaw Measurements

(d) $v$ Controls

(e) $w$ Controls

Figure 4.18: The system measurements and controls, captured alongside the image sequence of Figure 4.17. The dip in platform velocities at image 21 is due to the sustained loss of a target resulting from motion blur.

In the human-following system presented here, the primary causes of motion blur are sharp movements of the pan-tilt unit and abrupt platform accelerations. These movements form part of the controller response to a rapidly

moving target, so the need to minimise these motions by ramping up speeds results in a decrease in the allowable speeds which can be tracked.

Technically, the limitations of motion blur are on platform accelerations, and the speed should only be limited to the platform maximum, provided the time taken to reach this speed is long enough, but in practice this translates to a speed limitation. This is especially true in the case of walking humans, where accelerations are not easily controlled.

Despite limiting the system's speed of operation, the effects of motion blur do not affect the conclusions made in this work. Analysis of Figure 4.12 in Section 4.4.1.2 shows that the hybrid gain-scheduling controller still offers better performance than both the direction-based controller and point-to-point controller in the practical system's range of operation.

# Chapter 5

# Conclusions and Recommendations

## 5.1 Conclusions

This thesis has presented the design of the control and vision components for use in a monocular vision-based human-following robot. Traditional approaches to human-following typically involve a controller that causes platforms to navigate directly towards targets, but this work has argued that better following performance can be obtained through the use of a controller that incorporates target orientation information. This work has answered the following research questions.

- Is there any benefit in direction-based control over point-to-point control for a generic target follower?
- If the benefits are negligible, is it possible to use orientation information in such a way as to enhance them?
- If so, is there a measure of human pose or orientation that makes it possible to incorporate these benefits into a human-following system?

The investigations conducted here have shown that both point-to-point and direction-based controllers have benefits that are exhibited in complementary regions of operation. Although a purely direction-based controller suffers from various limitations, this thesis has shown that a hybrid gain-scheduling combination of two traditional controllers, incorporating target orientation information, offers better target-following performance than its components.

In the case of human following the inclusion of target orientation information requires that a definition and means of estimating a human's orientation be available. This work has presented a human orientation measure that has been shown to be suitable for the purposes of wheeled platform control. Experimental results have been provided to show that the hybrid controller

incorporating this measure of operation provides better performance than the individual point-to-point and direction-based controllers it is comprised of.

### 5.1.1 Visual Servo Control Strategy

Investigations into the various forms of vision-based control showed that two types of visual servo controllers are typically used: position-based visual servo (PBVS) and image-based visual servo techniques (IBVS). IBVS methods perform control in the image plane, using calculations based entirely on pixel coordinates, while PBVS techniques separate the control into a pose estimation and control phase. Simulations were presented to determine the suitability of IBVS approaches to a target-following application, given the known benefits of PBVS methods. The simulations showed that IBVS methods can result in undesirable camera motions, as control is only performed in the image plane. For this reason, a PBVS technique was selected for use in this work.

### 5.1.2 Visual Target Detection

Before control strategies could be implemented, however, a suitable means of performing target recognition was required. Various approaches to target recognition were identified and examined, with the decision made to use a feature-based recognition strategy. Although feature-based recognition systems require relatively salient targets, these techniques are effective at recognising objects in cluttered environments. The greatest benefit to feature-based approaches, however, is that the feature position outputs of these techniques are suitable for use in pose estimation algorithms.

Three feature-based recognition techniques were selected from the literature as potential approaches for this system, the SURF feature matching scheme, a Ferns-based classifier and the Kanade Lucas tracker. These approaches were selected as candidates for use in the system, given their reported speed. The processing rate of the object recognition system is extremely important in this context, as it involves real-time control.

Implementations of these algorithms were used to detect targets in a dataset containing images of targets captured under conditions likely to occur in an actual target-following task. The algorithms were compared using five criteria: speed, recognition rate, match accuracy, the range of detectable yaw motions and the range of detectable scales. Though slower than the other two algorithms, the SURF matching scheme was selected for use in this system due to its ability to detect targets accurately over a wider range of viewpoints and scales.

Improvements in the speed of the SURF algorithm were obtained by implementing a windowing process, where only a region of interest in an image is searched for targets. The region of interest is predicted based on prior detections. In addition, resolution adjustments were used, where the image

resolution used to detect targets was scaled up and down, depending on the size of targets in an image. An efficient implementation of the SURF matching scheme, operating at approximately 12 fps, was obtained by applying these improvements.

### 5.1.3 Human Pose Estimation using Monocular Vision

This thesis proposed that the pose of a walking person's upper torso typically indicates their travelling direction, and that a simple planar fit to the back of the torso contains sufficient information to infer travelling direction. Fitting a plane through detected feature points on a target is trivial if the 3D locations of the features are known, but only 2D image coordinates are available in this system, since only a single camera is used.

This work showed that by estimating the homography between two views of a human torso in a robust manner, the rotation and translation between two planar approximations of the torso could be obtained. This RANSAC-based robust estimation provided a valuable means of estimating the uncertainty in a decomposition, a weighting based on the ratio of inliers used in the estimate to the total number of detectable features. Experimental results using a model of the SURF feature-based target detector showed that this measure was closely correlated to the error in an estimate.

The homography can be decomposed into rotation and translation parameters through the use of singular value decomposition. This process is somewhat lengthy, and this work showed that it could be simplified dramatically by intelligently selecting a training or template image in a fronto-parallel configuration.

Experimental results also showed the efficacy of the human pose estimate, with practical measurements proving that the estimate was conceptually correct. Although the accuracy of the pose estimate could not be proven, the thesis showed that the accuracy is of little consequence for the purposes of control, as the measure of all parameters of interest is monotonic within the region of operation.

### 5.1.4 Tracking to Improve Estimates

As the pose estimate obtained is noisy, filtering action is required before the measure can be used by a suitable control system. Exponentially weighted filtering action was desired, but a means of including the certainty in a measurement in the filter was required. A Kalman filter was used to accomplish this. The thesis showed that the Kalman filter performed as desired, and was particularly effective at predicting target motion when momentary target losses occurred due to motion blur.

### 5.1.5 Controller Selection

Initially, this work identified two controllers of interest for use in the human-following system. The first, most commonly used in human-following systems, was a point-to-point controller that navigates directly to a target, without considering relative target orientation. While the point-to-point controller is able to follow rapidly moving targets, it is unable to follow sharply turning objects with little forward velocity. An alternative to this controller is a direction-based controller that follows targets by minimising both relative orientation and position errors. This controller is less susceptible to losing targets that are sharply turning, but experiences difficulties when following rapid targets.

In an attempt to combine the benefits of each of these controllers, a hybrid gain-scheduling controller was developed. This controller phases between the point-to-point and direction-based controller, depending on the magnitude of the target orientation. Monte Carlo simulations using a model of the feature-based target detection system showed that the hybrid controller dramatically improves the following performance of both controllers, only experiencing difficulties in following sharply turning targets at low forward velocities. This indicates that the hybrid controller should be better equipped to deal with human motion.

This was confirmed by an actual human-following trial, which showed that the hybrid gain-scheduling controller outperformed its components. In all experiments, our performance measure was the ability to follow a target without losing sight of it.

### 5.1.6 Implementation

A commercially available Pioneer P3-AT mobile platform was used as a base for the integration of the software components presented here and is controlled via serially transmitted actuation commands. Software was designed in C++ using the Open Computer Vision software libraries and an open source robotic architecture (Candy *et al.* (2010)) developed by the Council for Scientific and Industrial Research's Mobile Intelligent Autonomous Systems Group. The software architecture allows for easier message passing and modular node-based design using a publish/subscribe framework. Experiments were conducted on a Dell Latitude 6400 dual-core notebook with 2 GB RAM. A Prosilica GigE Ethernet camera was mounted on a commercially available pan-tilt unit, that accepts pan and tilt commands through serial communications.

This thesis discussed the theoretical limitations of a human-following system, which are primarily based on the camera and lens used, together with constraints on the allowable platform motion. Experimentation showed that the practical limitations were lower than that theoretically achievable, due to motion blur introduced in the system by rapidly moving targets. Motion blur can be countered through better control of lighting conditions, by purchasing

better cameras, or through various software algorithms. This is beyond the scope of this work however, and the system limitations introduced by motion blur do not affect the conclusions of this work.

## 5.2 Recommendations

The work has presented a position-based visual servo approach to human following. A definition and means of measuring human pose has been provided and shown to be suitable for the purposes of controlling a mobile platform. A hybrid gain-scheduling controller utilising this pose estimate was developed and shown to perform better than two standard controllers, traditionally used for wheeled platform navigation.

Note that this work was control oriented, and as a result did not involve the use of more intelligent navigation schemes that may include collision avoidance. The conclusions of this work are still of use for planning-based navigation schemes however, as navigation schemes taking the constraints of the vision system modelled here into account could be used to provide improved human-following trajectories.

This work showed that motion blur greatly affected the feature-based target detection algorithm. A second detection scheme, using techniques less susceptible to motion blur, such as region-based tracking, could be combined with the feature-based approach presented here. While the region-based approach does not allow for pose estimation algorithms, it can be fused with a feature-based technique to allow for greater redundancy in target detection.

The computational load of vision-based algorithms is extremely large, and could be alleviated through the addition of a second laser-based tracking system. Laser-based tracking systems are extremely fast, and capable of providing accurate position measurements. These systems experience difficulties in discriminating between targets, so are suited to operating in conjunction with visual target recognition approaches.

The combination of recognition strategies will eliminate the detection induced problems of human-following to a large extent, but their fusion would require a better model of human motion. A model of human motion would benefit greatly from the human pose estimate presented in this work, which has been shown to contain suitable information for use in a human-following robot. Incorporating a suitable model of human motion into a human-following system would allow for improved object tracking and, together with the added redundancy of multiple target recognition strategies, allow for a robust and efficient human-following system.

# Appendices

# Appendix A

# Computer Vision Fundamentals

An introduction to perspective camera geometry and camera calibration is provided here. Initially, the pinhole camera model is introduced, along with homogeneous coordinates and required notation. Thereafter, camera calibration is discussed, with reference to finding both pinhole camera model parameters and lens distortion information.

## A.1 Pinhole Camera Approximation

Figure A.1 shows how an image is formed in an ideal pinhole camera, with point $\mathbf{m} = [x, y]^{\mathrm{T}}$ the projection of 3D world coordinate $\mathbf{M} = [X, Y, Z]^{\mathrm{T}}$ on the image plane. The image plane is centred on the principal or optical axis of the camera, a distance $f$ from the optical or camera centre. Using similar triangles, it can be shown that the image plane coordinates $[x, y]^{\mathrm{T}}$ are equivalent to the coordinates $[fX/Z, fY/Z]^{\mathrm{T}}$.

A homogeneous representation of points is more convenient for pose estimation calculations, given the perspective nature of the cameras considered here. A point with image coordinates $x$ and $y$, can be thought of as a point on a perspective line denoted by a family of homogeneous vectors $[kx, ky, k]^{\mathrm{T}}$. The projection of these points on the image plane has homogeneous coordinates $[x, y, 1]^{\mathrm{T}}$. When the world and image points are represented by homogeneous vectors, the camera projection can be expressed as a linear mapping using matrix multiplication:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{A.1.1}$$

Frequently, cameras have unequal pixel sizes, and the focal length differs in the vertical and horizontal directions. In addition, the image plane is typically offset from the principal axis. The camera projective matrix, denoted by $\mathbf{P}$, is

adjusted as follows:

$$\mathbf{P} = \begin{bmatrix} f_x & 0 & x_0 & 0 \\ 0 & f_y & y_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{A.1.2}$$

In general, $x_0$ and $y_0$ are the coordinates of the image centre.

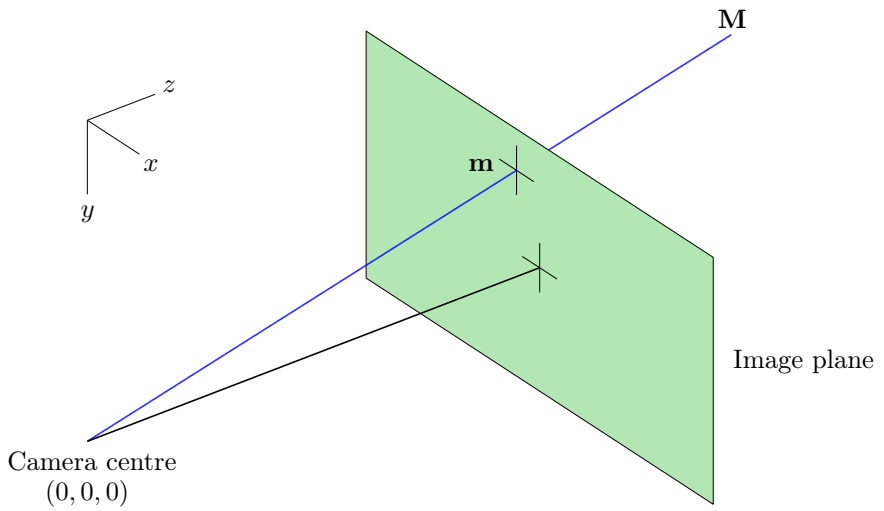



Figure A.1: The pinhole camera approximation. Point $\mathbf{m} = [x, y]^{\mathrm{T}}$ is the projection of 3D world coordinate $\mathbf{M} = [X, Y, Z]^{\mathrm{T}}$ onto the image plane.

The preceding definition of the pinhole camera model has been expressed in the camera coordinate frame. Frequently, however, points are expressed in some other coordinate frame and it is more convenient to express the camera projection matrix more generally as

$$\mathbf{P} = \mathbf{K}\,[\mathbf{R}|\mathbf{t}]\,. \tag{A.1.3}$$

Here, $\mathbf{t}$ represents a $3 \times 1$ translation vector that shifts the world coordinate axes onto the camera coordinate axes, and $\mathbf{R}$ is a $3 \times 3$ rotation matrix that rotates the world coordinate axes into alignment with the camera coordinate axes. These matrices are collectively termed the extrinsic camera effects. The matrix $\mathbf{K}$, calculated as

$$\mathbf{K} = \begin{bmatrix} f_x & 0 & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{A.1.4}$$

is referred to as the camera calibration or intrinsic camera matrix and encompasses the internal effects of the camera.

The camera calibration matrix is required by most pose estimation algorithms and can be obtained through the process of camera calibration described in Section A.2. It is important to note that the pinhole camera approximation does not consider camera lens distortion and requires that images be de-warped before it can be used. Image de-warping is generally a bi-product of camera calibration and is also discussed in Section A.2.

## A.2 Camera Calibration

The objective of camera calibration is to find the intrinsic camera matrix, **K**, through the analysis of multiple views of a calibration object. The Open Computer Vision C++ library (OpenCV) provides built-in routines to accomplish this, so only a short description of their approach is provided here.

More information regarding OpenCV and the fundamental computer vision algorithms it makes use of can be found in the OpenCV inspired textbook by Bradski and Kaehler (2008).

The OpenCV calibration routines require multiple images of a planar checkerboard (Figure A.2), with blocks of known dimensions to solve for the intrinsic camera properties.

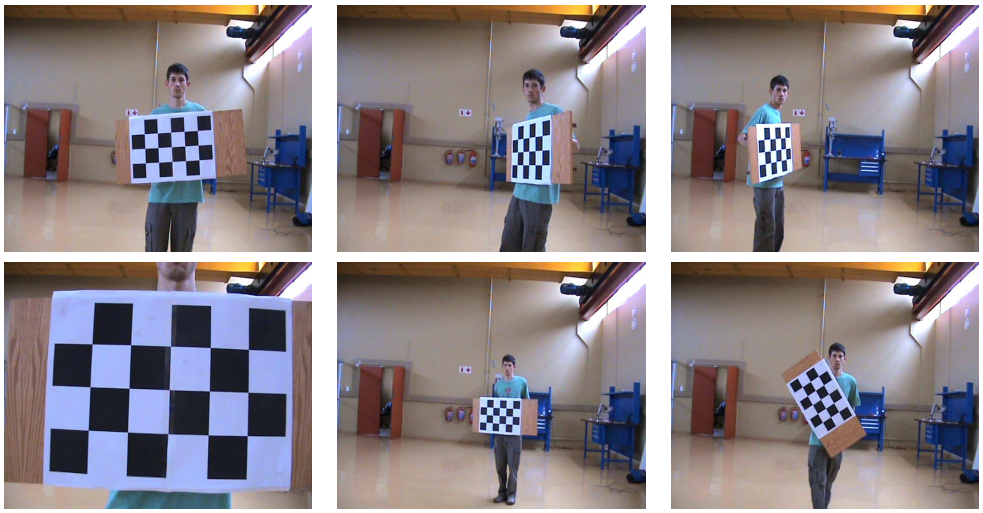


Figure A.2: Multiple images of a planar checkerboard are used to determine the intrinsic camera parameters in the pinhole camera model.

Initially, the corners of the checkerboard are accurately located using subpixel refinement. The 3D locations of these corners, $[X, Y, Z]^{\mathrm{T}}$, are mapped to

the image plane coordinates, $[x, y]^{\mathrm{T}}$, by the relationship

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda \mathbf{K} \left[\mathbf{R}|\mathbf{t}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \lambda \mathbf{K} \left[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{r_3}\ \mathbf{t}\right] \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{A.2.1}$$

Here, $\mathbf{K}$ is the desired camera calibration matrix, $\lambda$ a scale factor, $\mathbf{t}$ a translation and $\mathbf{R}$ a rotation with column vectors $\mathbf{r_1}$, $\mathbf{r_2}$ and $\mathbf{r_3}$. The object plane can be redefined such that $Z = 0$, without loss of generality. Then,

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \lambda \mathbf{K} \left[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{r_3}\ \mathbf{t}\right] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \lambda \mathbf{K} \left[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{t}\right] \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}. \tag{A.2.2}$$

### A.2.1 Calibration

The $3 \times 3$ matrix mapping 3D coordinates on the object plane to the image plane is termed a homography, which maps all coplanar features to the image plane. This homography,

$$\hat{\mathbf{H}} = \left[\mathbf{h_1}\ \mathbf{h_2}\ \mathbf{h_3}\right] = \lambda \mathbf{K} \left[\mathbf{r_1}\ \mathbf{r_2}\ \mathbf{t}\right], \tag{A.2.3}$$

can be calculated using the direct linear transform (Appendix C.1), given a set of corresponding source and image points. From these column vector equations, we obtain

$$\mathbf{r_1} = \frac{1}{\lambda}\mathbf{K}^{-1}\mathbf{h_1}, \quad \mathbf{r_2} = \frac{1}{\lambda}\mathbf{K}^{-1}\mathbf{h_2}, \quad \mathbf{t} = \frac{1}{\lambda}\mathbf{K}^{-1}\mathbf{h_3}. \tag{A.2.4}$$

Since $\mathbf{R}$ is a rotation, with scale extracted, its column vectors are orthonormal and

$$\mathbf{r_1}^{\mathrm{T}}\mathbf{r_2} = 0. \tag{A.2.5}$$

After substitution, this constraint becomes

$$\mathbf{h_1}^{\mathrm{T}}\mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h_2} = 0. \tag{A.2.6}$$

The norms of the rotation matrix column vectors are equal, which implies that

$$\mathbf{r_1}^{\mathrm{T}}\mathbf{r_1} = \mathbf{r_2}^{\mathrm{T}}\mathbf{r_2}. \tag{A.2.7}$$

This gives us a second constraint,

$$\mathbf{h_1}^{\mathrm{T}}\mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h_1} = \mathbf{h_2}^{\mathrm{T}}\mathbf{K}^{-T}\mathbf{K}^{-1}\mathbf{h_2}. \tag{A.2.8}$$

Setting $\mathbf{B} = \mathbf{K}^{-T}\mathbf{K}^{-1}$, we find that

$$\mathbf{B} = \begin{bmatrix} \frac{1}{f_x^2} & 0 & -\frac{x_0}{f_x^2} \\ 0 & \frac{1}{f_y^2} & -\frac{y_0}{f_y^2} \\ -\frac{x_0}{f_x^2} & -\frac{y_0}{f_y^2} & \frac{x_0^2}{f_x^2} + \frac{y_0^2}{f_y^2} + 1 \end{bmatrix}. \tag{A.2.9}$$

Given the constraints of Equations A.2.6 and A.2.8, and at least two planar homographies, we can solve for $\mathbf{B}$. It is then a simple matter to extract the intrinsic camera parameters from $\mathbf{B}$. Although this process finds a calibration matrix, the effects of camera lens distortions are not yet compensated for.

### A.2.2 Distortion Modelling

In practice, the pinhole camera model of Section A.1 is invalid and needs to be amended to consider lens distortion. Two primary forms of distortion are considered, radial and tangential distortion. Modelling of these distortion types follows the work of Brown (1971).

Radial distortion causes a bulging of pixels near the edge of an image, but is generally small near an image centre. As a result radial distortion can be approximated by the first few terms of a Taylor series expansion around $r = 0$, where r is the radial distance between a pixel and the (calibrated) image centre. This approximation is combined with a tangential distortion model characterised by two distortion parameters, $p_1$ and $p_2$ to give

$$\begin{aligned} x_{\text{corrected}} &= x\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ &+ 2p_1 xy + p_2\left(r^2 + 2x^2\right) \qquad \text{(A.2.10)} \\ y_{\text{corrected}} &= y\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) \\ &+ 2p_2 xy + p_1\left(r^2 + 2y^2\right). \qquad \text{(A.2.11)} \end{aligned}$$

Here, $x$ and $y$ are the original distorted pixel locations, with $x_{\text{corrected}}$ and $y_{\text{corrected}}$ the corrected locations. $k_1$, $k_2$ and $k_3$ are the radial distortion coefficients. The five distortion coefficients are combined into a single ordered distortion vector, $[k_1, k_2, p_1, p_2, k_3]^{\text{T}}$.

The distortion coefficients are solved for during the calibration process by projecting the 3D calibration object points onto the image plane, using the estimated intrinsic and extrinsic calibration matrices. These points represent perfect or corrected pinhole camera image plane coordinates. Substitution of these projected points, and the original points detected, into Equations A.2.10 and A.2.11 allows for the distortion coefficients to be estimated.

Correction of the position of image pixels through de-warping, using these distortion coefficients, is crucial to the operation of the pose estimation algorithms.

## A.3 Pyramidal Scale Space

The concept of scale in an image is easily understood. As objects in an image are viewed from further away, the level of visible detail is reduced. This reduction in detail can be imitated by a scale-space representation of an image, described in detail by Lindeberg (1994), which is obtained by convolving the image with a set of Gaussian kernels. The scale-space representation of an image is defined as

$$L(x, y; \sigma) = (g(., .; \sigma) * I(., .))(x, y) \tag{A.3.1}$$

where $I$ denotes the image, $x$ and $y$ image coordinates, and $\sigma$ the scale of the image. The symbol $*$ denotes the convolution operation. The Gaussian kernel is given by

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma} e^{\frac{-\left(x^2+y^2\right)}{2\sigma}}. \tag{A.3.2}$$

In the limit, for $\sigma = 0$, the Gaussian kernel becomes an impulse function, and the convolution result is the original image. Figure A.3 shows the detail reduction in a scale operation. Feature detectors are easily extended to provide scale invariance, by operating on multiple images in scale-space. This does occur at the expense of computational time, though, so in practice a limited number of discrete scale levels are used.


Figure A.3: Scaling an image removes detail and simulates the appearance of the areas around features at distance.

Since detail is removed in the smoothing or scaling step, less resolution is needed to store the relevant information. Additional computational savings are made if the scaled image is also sub-sampled. A discrete scale-space

representation of images that are sub-sampled is referred to as a pyramidal representation.

Figure A.4 shows a sample discrete scale space representation of an image of a mobile robot. The size of the image decreases with each scale value, incorporating the redundancy in resolution resulting from a reduction of image detail.

Figure A.4: An example of a scale-space pyramid created by consecutive smoothing and down-sampling of an image.

# Appendix B

# 3D Pose Estimation

## B.1 Procrustes Orthogonal Analysis

Consider a set of $n$ column vector pairs, $(\mathbf{x}_1, \mathbf{y}_1)\,;\ (\mathbf{x}_2, \mathbf{y}_2)\,;\ \ldots;\ (\mathbf{x}_n, \mathbf{y}_n)$, with each column containing the coordinates of an individual feature. Schönemann (1966) showed that the corresponding feature coordinates in a pair can be related by a scale $\lambda$, rotation $\mathbf{R}$ and translation $\mathbf{t}$ such that

$$\mathbf{x}_i = \lambda \mathbf{R} \mathbf{y}_i + \mathbf{t} \quad \text{for all } i = 1, 2, \ldots, n, \tag{B.1.1}$$

and provided a means of calculating these matrices, which maintain Euclidean distance constraints between feature coordinates. Initially, features are grouped into point clouds, shifted by their centroids and normalised. Let

$$\mathbf{X} = [\mathbf{x}_1 - \overline{\mathbf{x}}, \mathbf{x}_2 - \overline{\mathbf{x}}, \ldots, \mathbf{x}_n - \overline{\mathbf{x}}] \tag{B.1.2}$$

$$\mathbf{Y} = [\mathbf{y}_1 - \overline{\mathbf{y}}, \mathbf{y}_2 - \overline{\mathbf{y}}, \ldots, \mathbf{y}_n - \overline{\mathbf{y}}]\,, \tag{B.1.3}$$

with $\overline{\mathbf{x}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{x_i}$ and $\overline{\mathbf{y}} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{y_i}$. Then

$$\mathbf{X_n} = \frac{\mathbf{X}}{||\,\mathbf{X}\,||_{\mathrm{F}}}, \quad \mathbf{Y_n} = \frac{\mathbf{Y}}{||\,\mathbf{Y}\,||_{\mathrm{F}}}, \tag{B.1.4}$$

with $||\,.\,||_{\mathrm{F}}$ the Frobenius norm.

Thereafter, Singular Value Decomposition (SVD) of the matrix $\mathbf{A} = \mathbf{Y}_n \mathbf{X}_n^{\mathrm{T}}$ is completed:

$$\mathbf{A} = \mathbf{L}\ \mathbf{D}\ \mathbf{M}^{\mathrm{T}}. \tag{B.1.5}$$

Using this result, the rotation and scale parameters can be calculated:

$$\mathbf{R} = \mathbf{M}\mathbf{L}^{\mathrm{T}}, \tag{B.1.6}$$

$$\lambda = \mathrm{trace}\,(\mathbf{D})\,\frac{||\,\mathbf{X}\,||_{\mathrm{F}}}{||\,\mathbf{Y}\,||_{\mathrm{F}}}. \tag{B.1.7}$$

The translation matrix is then easily obtained by

$$\mathbf{t} = \overline{\mathbf{x}} - \lambda \mathbf{R} \overline{\mathbf{y}}. \tag{B.1.8}$$

# Appendix C

# Homographies

## C.1 Direct Linear Transform

The normalised direct linear transform (DLT) is an algorithm, presented by Hartley and Zisserman (2004), which is used to compute the 2D homography between coplanar point correspondences in two image scenes:

$$\mathbf{x}_1 = \mathbf{H}\ \mathbf{x}_2. \tag{C.1.1}$$

The homography is obtained using a set of point correspondences. Initially, each point set is normalised by computing a transformation $\mathbf{T}$ that shifts the centroid to point $(0, 0)$ and scales point coordinates such that their average distance from the centroid is $\sqrt{2}$. This normalisation ensures that the equations used later are well conditioned. For a single normalised correspondence pair, $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$,

$$\tilde{\mathbf{x}}_2 = \tilde{\mathbf{H}}\ \tilde{\mathbf{x}}_1 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}. \tag{C.1.2}$$

In inhomogeneous coordinates, or the actual image plane positions $\hat{x}_2 = \frac{x_2}{z_2}$ and $\hat{y}_2 = \frac{y_2}{z_2}$, we find

$$\hat{x}_2 = \frac{h_{11}x_1 + h_{12}y_1 + h_{13}z_1}{h_{31}x_1 + h_{32}y_1 + h_{33}z_1}, \quad \hat{y}_2 = \frac{h_{21}x_1 + h_{22}y_1 + h_{23}z_1}{h_{31}x_1 + h_{32}y_1 + h_{33}z_1}. \tag{C.1.3}$$

$z_1$ is unknown and cannot be determined using a single camera, which results in the loss of absolute scale. Rearranging, with $z_1 = 1$, gives

$$\hat{x}_2\,(h_{31}x_1 + h_{32}y_1 + h_{33}) = h_{11}x_1 + h_{12}y_1 + h_{13} \tag{C.1.4}$$

$$\hat{y}_2\,(h_{31}x_1 + h_{32}y_1 + h_{33}) = h_{21}x_1 + h_{22}y_1 + h_{23}. \tag{C.1.5}$$

These equations can be rearranged into a linear least squares form $\mathbf{Ah} = \mathbf{0}$, where

$$\mathbf{h} = [h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33}]^{\mathrm{T}} \quad \text{(C.1.6)}$$

$$\mathbf{A} = \begin{bmatrix} -x_1 & -y_1 & -1 & 0 & 0 & 0 & \hat{x}_2 x_1 & \hat{x}_2 y_1 & \hat{x}_2 \\ 0 & 0 & 0 & -x_1 & -y_1 & -1 & \hat{y}_2 x_1 & \hat{y}_2 y_1 & \hat{y}_2 \end{bmatrix}. \quad \text{(C.1.7)}$$

The homography is then easily obtained by stacking four or more non-co-linear point correspondences in this form. Finally, the de-normalised homography is obtained by removing the normalisation transformation as follows:

$$\mathbf{H} = \mathbf{T_1}^{-1}\, \tilde{\mathbf{H}}\, \mathbf{T_2}. \quad \text{(C.1.8)}$$

## C.2 Homography Decomposition

Once the homography has been determined, the various pose parameters mapping the current camera coordinate system to the desired (template) camera coordinate system can be retrieved from the decomposition of Faugeras and Lustman (1988):

$$\mathbf{H} = \mathbf{K}\left(\mathbf{R} + \mathbf{t}\,\mathbf{n}^T\right)\mathbf{K}^{-1}. \quad \text{(C.2.1)}$$

Here, $\mathbf{K}$ is the intrinsic camera calibration matrix, $\mathbf{R}$ a rotation matrix, $\mathbf{t}$ the translation of the camera and $\mathbf{n}$ a vector normal to the planar target. There are eight degrees of freedom: three in the rotation and five in the surface normal and camera translation (which is extractable up to scale).

The algorithm development in the work by Faugeras and Lustman (1988) is closely followed here. Initially, camera effects are removed from $\mathbf{H}$ and the singular value decomposition (SVD) of the result is obtained as

$$\widehat{\mathbf{H}} = \mathbf{K}^{-1}\,\mathbf{H}\,\mathbf{K} = \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^T. \quad \text{(C.2.2)}$$

The diagonal matrix $\mathbf{\Sigma}$, containing singular values of $\widehat{\mathbf{H}}$, can be decomposed into the various pose parameters, such that

$$\mathbf{\Sigma} = \tilde{d}\widetilde{\mathbf{R}} + \widetilde{\mathbf{t}}\,\widetilde{\mathbf{n}}^T. \quad \text{(C.2.3)}$$

Here, $\tilde{d}$ is a scalar, $\widetilde{\mathbf{R}}$ is a $3 \times 3$ rotation matrix, $\widetilde{\mathbf{t}}$ is a $3 \times 1$ translation vector and $\widetilde{\mathbf{n}}$ is a $3 \times 1$ normal vector.

The final decomposition elements of $\mathbf{H}$ are then calculated according to

$$\mathbf{R} = s\mathbf{U}\,\widetilde{\mathbf{R}}\,\mathbf{V}^T \quad \text{(C.2.4)}$$

$$\mathbf{t} = \mathbf{U}\,\widetilde{\mathbf{t}} \quad \text{(C.2.5)}$$

$$\mathbf{n} = \mathbf{V}\,\widetilde{\mathbf{n}} \quad \text{(C.2.6)}$$

$$d = s\tilde{d} \quad \text{(C.2.7)}$$

$$s = \det\mathbf{U}\,\det\mathbf{V}. \quad \text{(C.2.8)}$$

Representing vectors in a canonical basis $(\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3)$, with $\tilde{\mathbf{n}} = x_1\mathbf{e}_1 + x_2\mathbf{e}_2 + x_3\mathbf{e}_3$, leads to three vector equations:

$$\mathbf{\Sigma}_i\mathbf{e}_i = \tilde{d}\tilde{\mathbf{R}}\mathbf{e}_i + \tilde{\mathbf{t}}x_i \quad \text{for } i = 1, 2, 3. \tag{C.2.9}$$

Here, basis $\mathbf{e}_i$ effectively extracts the $i$th column vector from each matrix. Removing $\tilde{\mathbf{t}}$ results in three more vector equations:

$$\tilde{d}\tilde{\mathbf{R}}\left(x_j\mathbf{e}_i - x_i\mathbf{e}_j\right) = \mathbf{\Sigma}_i x_j\mathbf{e}_i - \mathbf{\Sigma}_j x_i\mathbf{e}_j \quad \text{for } i \neq j. \tag{C.2.10}$$

At this point, we should note that $\tilde{\mathbf{n}}$ has a unit norm, since $\mathbf{V}$ is orthogonal and preserves the vector norm of $\mathbf{n}$. $\tilde{\mathbf{R}}$ is a rotation and by definition has orthogonal column vectors with unit norms. Squaring both sides of this equation, and summing the rows of each vector equation thus removes the rotation parameters, resulting in a linear system with unknowns $x_1^2,\ x_2^2$ and $x_3^2$:

$$\begin{aligned}
\left(\tilde{d}^2 - \sigma_2^2\right)x_1^2 + \left(\tilde{d}^2 - \sigma_1^2\right)x_2^2 &= 0 \\
\left(\tilde{d}^2 - \sigma_3^2\right)x_2^2 + \left(\tilde{d}^2 - \sigma_2^2\right)x_3^2 &= 0 \\
\left(\tilde{d}^2 - \sigma_1^2\right)x_3^2 + \left(\tilde{d}^2 - \sigma_3^2\right)x_1^2 &= 0.
\end{aligned} \tag{C.2.11}$$

$\sigma_1,\ \sigma_2$ and $\sigma_3$ are the singular values of $\widehat{\mathbf{H}}$ or the diagonal elements of $\mathbf{\Sigma}$, with $\sigma_1 \geq \sigma_2 \geq \sigma_3$. We require a non-zero solution, so the determinant of this system must be zero:

$$\left(\tilde{d}^2 - \sigma_1^2\right)\left(\tilde{d}^2 - \sigma_2^2\right)\left(\tilde{d}^2 - \sigma_3^2\right) = 0. \tag{C.2.12}$$

Faugeras and Lustman (1988) proceed to show that the only solution to this determinant equation is $\tilde{d} = \pm\sigma_2$, as solutions $\tilde{d} = \pm\sigma_1$ and $\tilde{d} = \pm\sigma_3$ fail to preserve the vector norm $\tilde{\mathbf{n}}$. Solving equation C.2.11, including the unity norm constraint $x_1^2 + x_2^2 + x_3^2 = 1$, leads to a solution for $\tilde{\mathbf{n}}$:

$$\begin{aligned}
x_1 &= \pm\sqrt{\frac{\sigma_1^2 - \sigma_2^2}{\sigma_1^2 - \sigma_3^2}} \\
x_2 &= 0 \\
x_3 &= \pm\sqrt{\frac{\sigma_2^2 - \sigma_3^2}{\sigma_1^2 - \sigma_3^2}}.
\end{aligned} \tag{C.2.13}$$

The solution development now needs to be separated into two cases, $\tilde{d} > 0$ and $\tilde{d} < 0$. We start with the case $\tilde{d} > 0$.

Given that $x_2 = 0$, analysis of equation C.2.10 when $i = 2$ shows that $\tilde{\mathbf{R}}\mathbf{e}_2 = \mathbf{e}_2$. This means that $\tilde{\mathbf{R}}$ is a rotation about axis $\mathbf{e}_2$ and hence can be computed as

$$\tilde{\mathbf{R}} = \begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix}. \tag{C.2.14}$$

Using equations C.2.10 and C.2.14, we find

$$\sin\theta = (\sigma_1 - \sigma_3)\frac{x_1 x_3}{\sigma_2} \tag{C.2.15}$$

$$\cos\theta = \frac{\sigma_1 x_3^2 + \sigma_3 x_1^2}{\sigma_2}. \tag{C.2.16}$$

Substitution into equation C.2.9 provides a solution for the translation,

$$\tilde{\mathbf{t}} = (\sigma_1 - \sigma_3)\begin{bmatrix} x_1 \\ 0 \\ -x_3 \end{bmatrix}. \tag{C.2.17}$$

If the singular values, $\sigma_i$, are equal, these equations are undefined and the motion is a pure rotation with no translation and an undefined normal.

Development of solutions for the case $\tilde{d} < 0$ follows much the same process as that just followed. Analysis of equation C.2.10 when $i = 2$ now shows that $\tilde{\mathbf{R}}\mathbf{e}_2 = -\mathbf{e}_2$ and the rotation becomes

$$\tilde{\mathbf{R}} = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & -1 & 0 \\ \sin\phi & 0 & -\cos\phi \end{bmatrix}. \tag{C.2.18}$$

Using equations C.2.10 and C.2.14, we find

$$\sin\phi = (\sigma_1 + \sigma_3)\frac{x_1 x_3}{\sigma_2} \tag{C.2.19}$$

$$\cos\phi = \frac{\sigma_3 x_1^2 - \sigma_1 x_3^2}{\sigma_2}. \tag{C.2.20}$$

Substitution into equation C.2.9 provides a slightly different solution for the translation,

$$\tilde{\mathbf{t}} = (\sigma_1 + \sigma_3)\begin{bmatrix} x_1 \\ 0 \\ x_3 \end{bmatrix}. \tag{C.2.21}$$

Once more, if the singular values are equal, these equations no longer hold as the homography represents pure rotation.

We can see that the algorithm provides eight different solutions but, fortunately, not all are physically possible. The solution set is immediately reduced to four by including the constraint that both image frames must be located on the same side of the target object or, in other words, that the object viewed cannot be transparent. A second constraint, enforcing that visible points must be in front of both cameras, reduces the set to two solutions. Finally a single solution is obtained by incorporating assumed knowledge of the surface normal in the desired view. In cases where no knowledge of the surface normal is available, more than one image from a moving camera can be used to select the most suitable solution.

# Appendix D

# Tracking Filters

## D.1 Extended Kalman Filter

The extended Kalman filter (EKF), McGee and Schmidt (1985), is a nonlinear extension to the Kalman filter, which allows for state estimation by linearising models about the current estimate. The extended Kalman filter operates under the assumption that measurement and prediction noise is Gaussian.

The EKF consists of two stages: prediction and update. In this context, a prediction of the future pose is made based on a motion model, $\mathbf{f}$, and then updated using information obtained from the pose measurement system. Given a measurement $\mathbf{z}_k$, the predicted state $\hat{\mathbf{x}}_{k|k-1}$ and predicted covariance $\mathbf{P}_{k|k-1}$ are

$$\hat{\mathbf{x}}_{k|k-1} = \mathbf{f}(\hat{\mathbf{x}}_{k-1|k-1}, \mathbf{u}_k) \tag{D.1.1}$$

$$\mathbf{P}_{k|k-1} = \mathbf{F}_k \mathbf{P}_{k-1|k-1} \mathbf{F}_k^T + \mathbf{Q}_k \tag{D.1.2}$$

with $\mathbf{F}_k$ a first-order linearisation of the system update equations $\mathbf{f}$, and $\mathbf{Q}_k$ the process noise covariance matrix. A subscript $k$ indicates the $k$-th sample, while a subscript $k|k-1$ indicates a quantity associated with sample $k$, but estimated using information from sample $k-1$. The measurement and covariance residuals are:

$$\tilde{\mathbf{y}}_k = \mathbf{z}_k - \mathbf{h}(\hat{\mathbf{x}}_{k|k-1}) \tag{D.1.3}$$

$$\mathbf{S}_k = \mathbf{H}_k \mathbf{P}_{k|k-1} \mathbf{H}_k^T + \mathbf{R}_k \tag{D.1.4}$$

with $\mathbf{H}_k$ a first-order linearisation of the measurement model $\mathbf{h}$ and $\mathbf{R}_k$ the measurement noise covariance matrix. Then, the updated state and covariance estimates are given by:

$$\hat{\mathbf{x}}_{k|k} = \hat{\mathbf{x}}_{k|k-1} + \mathbf{K}_k \tilde{\mathbf{y}}_k \tag{D.1.5}$$

$$\mathbf{P}_{k|k} = (\mathbf{I} - \mathbf{K}_k \mathbf{H}_k) \mathbf{P}_{k|k-1} \tag{D.1.6}$$

Here, $\mathbf{K}_k = \mathbf{P}_{k|k-1} \mathbf{H}_k^T \mathbf{S}_k^{-1}$ is the optimal Kalman gain for a linear system.

Using these equations, the operation of the EKF is easily understood. First, a prediction of the system state is made (Equation D.1.1), assuming that zero-mean process noise is present. An estimate of the uncertainty in this prediction is made by combining previous uncertainty with that introduced through control action, making allowances for disturbances (Equation D.1.2). A measurement is made and the uncertainty in prediction combined with the uncertainty in measurement (Equation D.1.4). Finally, a revised state estimate is obtained by an uncertainty weighted combination of prediction and measurement (Equation D.1.5).

# Appendix E

# 3D Camera Positioning

## E.1 Strapdown Inertial Navigation

The image-based visual servo control simulator of Section 3.1.1 requires that the 3D position of a 6 DOF camera be maintained, given a set of translational and rotational camera velocity inputs. This requires the application of various strapdown inertial navigation techniques, as the velocities are expressed in the camera body frame which moves with the control inputs.

Only the mathematics required to calculate camera position is presented here, with readers referred to the technical report of Woodman (2007) for further detail on inertial navigation systems.

We start by describing the attitude or orientation of the camera body frame relative to the global frame using a direction cosine matrix $\mathbf{C}$. Each column of this rotation matrix represents a unit vector along the body axes in terms of the global axes.

A translational velocity vector $\mathbf{v_c}$ in the camera body frame is equivalent to the vector

$$\mathbf{v_g} = \mathbf{C}\ \mathbf{v_c} \tag{E.1.1}$$

in the global frame. Our goal is to track the position and attitude of the camera over time. Figure E.1 illustrates the attitude tracking problem. The figure shows an angle axis representation of the camera velocities. $\omega_\mathbf{c}$ is the angular rotation about a velocity vector $\mathbf{v_c}$. The angle axis velocity representation can be decomposed into rotational and translational velocities about each of the body frame axes, a screw denoted by $[t_x,\ t_y,\ t_z,\ \omega_x,\ \omega_y,\ \omega_z]^{\mathrm{T}}$.

Using a small angle approximation, the assumption that there is very small change in attitude between time steps, the attitude update is given as

$$\dot{\mathbf{C}} = \mathbf{C}\ \mathbf{\Omega}, \tag{E.1.2}$$

where

$$\mathbf{\Omega} = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix}. \tag{E.1.3}$$

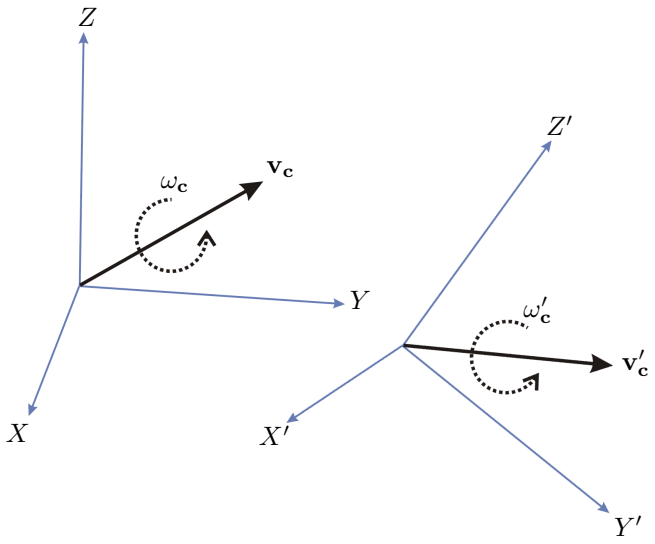



Figure E.1: The camera body frame undergoes translational and rotational velocities concurrently. In order to track the overall position and attitude of the camera, motion in the camera body frame needs to be shifted into a global reference frame.

The attitude of the camera is obtained by integrating this matrix over time. Unfortunately, the small angle approximation to attitude causes the rotation matrix to dilate over time. This is remedied to some extent by re-normalising the columns of the attitude matrix after each update. Camera position, $\mathbf{X} = [X,\ Y,\ Z]^{\mathrm{T}}$ is obtained by projecting each translational velocity component into the global reference frame,

$$\mathbf{v_g} = \mathbf{C}\ \mathbf{v_c} \tag{E.1.4}$$

$$\dot{\mathbf{X}} = \mathbf{v_g}, \tag{E.1.5}$$

and integrating over time.

The cumulative effect of small angle approximation errors leads to drift in position and attitude measurement. Fortunately, the drift does not affect the IBVS simulation results drastically, as only short distances are covered.

# List of References


Agarwal, A. and Triggs, B. (2004). 3D human pose from silhouettes by relevance vector regression. In: *Computer Vision and Pattern Recognition, Proceedings of the 2004 IEEE Computer Society Conference on*, vol. 2, pp. 882 – 888.

Amit, Y. and Geman, D. (1997). Shape quantization and recognition with randomized trees. *Neural Computation*, vol. 9, no. 7, pp. 1545–1588.

Anderson, F.C. and Pandy, M.G. (2001). Dynamic optimization of human walking. *Journal of Biomechanical Engineering*, vol. 123, no. 5, pp. 381–390.

Barron, J.L., Fleet, D.J., Beauchemin, S.S. and Burkitt, T.A. (1994). Performance of optical flow techniques. *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43–77.

Bay, H., Ess, A., Tuytelaars, T. and van Gool, L. (2008). Speeded-up robust features (SURF). *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359.

Beis, J.S. and Lowe, D.G. (1997). Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1000–1006.

Bradski, G. and Kaehler, A. (2008). *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, Inc.

Brown, D.C. (1971). Close-range camera calibration. *Photogrammetric Engineering*, vol. 37, no. 8, pp. 855–866.

Burke, M.G. (2010). Laser-based target tracking using principal component descriptors. In: *21st Symposium of the Pattern Recognition Association of South Africa*, 21, pp. 45 – 50.

Candy, L., Sabatta, D. and Claassens, J. (2010). Miasarch 0.1. Software architecture developed by Mobile Intelligent Autonomous Systems, Council for Scientific and Industrial Research, South Africa. "http://www.csir.co.za/mias".

Chaumette, F. (1998). Potential problems of stability and convergence in image-based and position-based visual servoing. In: Kriegman, D., Hager, G.. and Morse, A. (eds.), *The Confluence of Vision and Control*, pp. 66–78. Lecture Notes in Control and Information Science Series, No 237, Springer-Verlag.

Chaumette, F. and Hutchinson, S. (2008). *Springer Handbook of Robotics*, chap. 24: Visual Servoing and Visual Tracking, pp. 563–583. Springer.

Chen, Y.-R., Huang, C.-M. and Fu, L.-C. (2009). Upper body tracking for human-machine interaction with a moving camera. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 1917–1922.

Chen, Z. and Birchfield, S. (2007). Person following with a mobile robot using binocular feature-based tracking. In: *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 815–820.

Dai, W., Cuhadar, A. and Liu, P. (2008 23-26). Robot tracking using vision and laser sensors. In: *Automation Science and Engineering, 2008. CASE 2008. IEEE International Conference on*, pp. 169 –174.

Daniilidis, K. and Eklundh, J.-O. (2008). *Springer Handbook of Robotics*, chap. 23: 3-D Vision and Recognition, pp. 543–561. Springer.

Das, A.K., Fierro, R., Kumar, R.V., Ostrowski, J.P., Spletzer, J. and Taylor, C.J. (2002). A vision based formation control framework. *IEEE Transactions on Robotics and Automation*, vol. 18, no. 5, pp. 813–825.

Das, A.K., Fierro, R., Kumar, R.V., Southall, B., Spletzer, J. and Taylor, C.J. (2001). Real-time vison-based control of a nonholonomic mobile robot. In: *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*. IEEE.

David, P., DeMenthon, D., Duraiswami, R. and Samet, H. (2002). Softposit: Simultaneous pose and correspondence determination. In: *ECCV '02: Proceedings of the 7th European Conference on Computer Vision-Part III*, pp. 698–714. Springer-Verlag, London, UK.

DeMenthon, D.F. and Davis, L.S. (1995). Model-based object pose in 25 lines of code. *International Journal of Computer Vision*, vol. 15, pp. 123–141.

Faugeras, O. and Lustman, F. (1988). Motion and structure from motion in a piecewise planar environment. *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 2, pp. 485–508.

Ferrari, V., Marín-jiménez, M. and Zisserman, A. (2009). 2D human pose estimation in tv shows. In: *Proceedings of the Dagstuhl Seminar on Stastistical and Geometrical Approaches to Visual Motion Analysis*.

Fischler, M.A. and Bolles, R.C. (1987). *Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography*, pp. 726–740. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Germa, T., Lerasle, F., Ouadah, N., Cadenat, V. and Devy, M. (2009). Vision and RFID-based person tracking in crowds from a mobile robot. In: *IEEE/RSJ International Conference on Intelligent Robotics and Systems*, pp. 5591–5596.

Gevers, T. and Smeulders, A.W.M. (1999). Color-based object recognition. *Pattern Recognition*, vol. 32, no. 3, pp. 453 – 464.

Gockley, R., Forlizzi, J. and Simmons, R. (2007). Natural person-following behavior for social robots. In: *Proceedings of the ACM/IEEE International Conference on Human-Robot Interaction*, pp. 17–24.

Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In: *Proceedings of the fourth ALVEY Vision Conference*. ALVEY.

Hartley, R.I. and Zisserman, A. (2004). *Multiple View Geometry in Computer Vision*. 2nd edn. Cambridge University Press.

Hirai, N. and Mizoguchi, H. (2003). Visual tracking of human back and shoulder for person following robot. In: *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*, pp. 527–532.

Hutchinson, S., Hager, G.D. and Corke, P.I. (1996). A tutorial on visual servo control. *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670.

Kalman, R.E. (1960). 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.

Kass, M., Witkin, A. and Terzopoulos, D. (1988). Snakes: Active contour models. *International Journal of Computer Vision*, vol. 1, no. 4, pp. 321–331.

Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, vol. 2, pp. 506–513.

Latif, H., Sherkat, N. and Lotfi, A. (2009). Fusion of automation and teleoperation for person-following with mobile robots. In: *IEEE International Conference on Information and Automation*, pp. 1240–1245.

Lee, M. and Cohen, I. (2004). Human upper body pose estimation in static images. In: *European Conference on Computer Vision*, vol. 2, pp. 126–138.

Lepetit, V. and Fua, P. (2006). Keypoint recognition using randomized trees. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 1465–1479.

Levy, L.J. (1997 September). The KF: Navigation's integration workhorse. *GPS World*, pp. 65–71.

Lindeberg, T. (1994). Scale-space theory: A basic tool for analysing structures at different scales. *Journal of Applied Statistics*, pp. 224–270.

Lowe, D.G. (1999). Object recognition from local scale-invariant features. *Computer Vision, IEEE International Conference on*, vol. 2, p. 1150.

Lowe, D.G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, vol. 2, no. 60, pp. 91–110.

Lucas, B.D. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In: *IJCAI'81: Proceedings of the 7th international joint conference on Artificial intelligence*, pp. 674–679. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Ma, Y., Košecká, J. and Sastry, S. (1999). Vision guided navigation for a nonholonomic mobile robot. *IEEE Transactions on Robotics and Automation*, vol. 15, no. 3, pp. 521–536.

Ma, Y., Soatto, S., Kosecka, J. and Sastry, S. (2003). *An Invitation to 3-D Vision: From Images to Geometric Models*. SpringerVerlag.

Malis, E., Chaumette, F. and Boudet, S. (1999). 2 1/2 D Visual Servoing. *IEEE Transactions on Robotics and Automation*, vol. 15, pp. 238–250.

Maya-Mendez, M., Morin, P. and Samson, C. (2006). Control of a nonholonomic mobile robot via sensor-based target tracking and pose estimation. In: *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE/RSJ.

McGee, L.A. and Schmidt, S.F. (1985). *Discovery of the Kalman filter as a practical tool for aerospace and industry*. National Aeronautics and Space Administration, Ames Research Center, Moffett Field, California.

Mikolajczyk, K. and Schmid, C. (2002). An affine invariant interest point detector. In: *Proceedings of the European Conference on Computer Vision*, pp. 128–142. Springer Verlag.

Mikolajczyk, K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 10, pp. 1615–1630.

Morevec, H.P. (1977). Towards automatic visual obstacle avoidance. In: *IJCAI'77: Proceedings of the 5th International Joint Conference on Artificial Intelligence*, pp. 584–584. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Morin, P. and Samson, C. (2008). *Springer Handbook of Robotics*, chap. 34: Motion Control of Wheeled Mobile Robots, pp. 799–825. Springer.

Noble, A. (1989). *Descriptions of Image Surfaces*. Ph.D. thesis, Department of Engineering Science, Oxford University.

Ozuysal, M., Calonder, M., Lepetit, V. and Fua, P. (2010). Fast keypoint recognition using random ferns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, pp. 448–461.

Petrov, P. and Parent, M. (2006 Aug). Adaptive control for reversing a two-vehicle platoon. In: *Proceedings the 11th IFAC Symposium on Control in Transportation Systems (IFAC CTS2006)*. IFAC.

Plutarch (1914). *Plutarch. Lives Vol. I.* Translated by Perrin, Bernadotte. Loeb Classical Library Volume 46. Cambridge, MA. Harvard University Press. London. William Heinemann Ltd.

Quan, L. and Lan, Z. (1999). Linear n-point camera pose determination. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 774–780.

Ristic, B., Arulampalam, S. and Gordon, N. (2004). *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House.

Rosten, E. and Drummond, T. (2005). Fusing points and lines for high performance tracking. In: *IEEE International Conference on Computer Vision*, vol. 2, pp. 1508–1511.

Schlegel, C., Illmann, J., Jaberg, H., Schuster, M. and Worz, R. (1998). Vision based person tracking with a mobile robot. In: *British Machine Vision Conference*, pp. 418–427.

Schmid, C., Mohr, R. and Bauckhage, C. (1998). Comparing and evaluating interest points. In: *Proceedings of the 6th International Conference on Computer Vision, Bombay, India*. IEEE Computer Society Press.

Schönemann, P. (1966). A generalized solution of the orthogonal procrustes problem. *Psychometrika*, vol. 31, no. 1, pp. 1–10.

Shi, J. and Tomasi, C. (1994). Good features to track. *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR '94., 1994 IEEE Computer Society Conference on*, pp. 593–600.

Thrun, S., Burgard, W. and Fox, D. (2005). *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press.

Torre, V. and Poggio, T. (1984). On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, pp. 147–163.

Trajkovic, M. and Hedley, M. (1998). Fast corner detection. *Image and Vision Computing*, vol. 16, no. 2, pp. 75 – 87.

Vela, P., Betser, A., Malcolm, J. and Tannenbaum, A. (2009). Vision-based range regulation of a leader-follower formation. *IEEE Transactions on Control Systems Technology*.

Woodman, O.J. (2007). An introduction to inertial navigation. Tech. Rep. 696, University of Cambridge.

Yoon, Y., Kosaka, A. and Kak, A. (2008). A new Kalman-filter-based framework for fast and accurate visual tracking of rigid objects. *IEEE Transactions on Robotics*, vol. 24, no. 5, pp. 1238–1251.

Zhang, Z. (1992). Iterative point matching for registration of free-form curves. Research Report RR-1658, INRIA.

Zheng, Z., Wang, H. and Teoh, E.K. (1999). Analysis of gray level corner detection. *Pattern Recognition Letters*, vol. 20, no. 2, pp. 149–162.