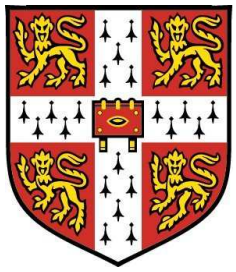
Visual Control of a Miniature Quad-Rotor Helicopter



Christopher Kemp

Churchill College

University of Cambridge

A thesis submitted for the degree of

Doctor of Philosophy

Declaration

I declare that this thesis is original and describes my own work. Where reference is made to the works of others the extent to which that work has been used is indicated and duly acknowledged in the text and bibliography. This dissertation is less than 60 000 words in length and no part of it has already or is currently being submitted for any other qualification at any other university.

Christopher Kemp

February 2006

Abstract

Automatically controlling the flight of an unmanned aerial vehicle has typically been achieved using a suite of sensors to give reliable estimates of the vehicle's position, orientation and velocity. Recently, on-board video cameras have been used as a primary source of these estimates, for the outdoor flight of large model helicopters. Here a much smaller, indoor model helicopter is automatically flown by employing just a single miniature transmitting camera. However, this task presents a stiff set of challenges to the visual tracking system used to obtain pose and velocity estimates from the transmitted images. This thesis shows how they can be overcome.

Tracking discontinuous motion using ambiguous image features means that it is not sufficient to consider just a single hypothesis for the camera pose. A method for efficiently generating a representation of a multi-modal posterior probability distribution is presented. The technique combines ideas from RANSAC and particle filtering such that the visual tracking problem can be partitioned into two levels, while maintaining multiple hypotheses throughout. The resulting system demonstrates a significant improvement in tracking reliability over previous unimodal approaches.

The use of multiple hypotheses however greatly increases the computational complexity of tracking and this causes difficulties when

operating in real-time. This is addressed using a technique for clustering measurements which simplifies the problem of high-dimensional parameter estimation. The key idea is to group measurements into clusters which are affected only by a subset of the parameter space. In contrast to static partitioning techniques, the method presented dynamically generates clusters at each step of the estimation. This substantially reduces the computation required, even for problems which cannot be partitioned in the traditional sense. Hence the resulting system is able to perform robust visual tracking of all six degrees of freedom in real-time.

Small helicopters are extremely unstable and require high bandwidth active control. This in turn requires accurate estimates of both the pose and (importantly) velocity of the helicopter. Unfortunately, visual tracking gives noisy estimates of position; this, coupled with the non-Gaussian dynamic processes, means that conventional filtering techniques either yield unusably poor velocity estimates or respond insufficiently to large disturbances. In this thesis, a multiple hypothesis filter which employs a dynamic programming algorithm is used. This filter is responsive, while yielding stable and accurate estimates of pose and velocity. These estimates can then be used with a simple controller to reliably control the orientation of the helicopter. This controller is in turn 'nested' inside a controller which chooses the orientation required to reach or hold at a desired position. This combined controller is able to reliably fly the helicopter and results from a variety of test flights are presented.

Acknowledgements

I sincerely thank my supervisor Dr. Tom Drummond and all those who made life in the MI lab so enjoyable. I am also indebted to my advisor, Prof. Roberto Cippola, and to Dr. Jim Henshaw, along with many others at Renishaw Plc.

The best bits of this thesis originated on mountain sides and while floating in the open sea. Many thanks to all those who dragged and accompanied me on such trips.

Contents

List of Figures

LIST OF FIGURES

Chapter 1

Introduction

An important application of visual tracking is to aid the automatic guidance of robots or autonomous vehicles. This thesis presents the development of several visual tracking algorithms which enable the computer control of a miniature aerial vehicle (MAV), in this case a model helicopter. One of the key requirements for the success of an automatic controller is an accurate estimate of the MAV's pose (position and orientation). In previous work with large model helicopters [e.g. Amidi 1996], this was obtained using a combination of GPS measurements, inertial sensing and visual information. The target of this research is the control of much smaller vehicles which, having greatly reduced payload capacities, are limited in the variety of sensors which can be carried. Further, the vehicle should operate in indoor environments where GPS measurements are unreliable. The aim is therefore to locate the vehicle using just the information from a miniature on-board video camera, which transmits images back to a base computer.

Visual tracking systems follow motion in a sequence of images from a video camera. If the video camera is moving through a static world, this sequence can be used to recover the motion of the camera. Motion between images may be tracked by placing artificial markers on the camera's surroundings; however, this limits the working range of the system and may be undesirable for aesthetic reasons. Recent research has concentrated on operating in un-instrumented areas,

by following the image motion of features which already exist in the scene.

Tracking the motion of a MAV in this way and in *real-time* is particularly challenging for several reasons. The primary difficulty is caused by the light weight and instability of the MAV, which mean that external disturbances have a large effect and the MAV's motion is relatively difficult to predict. Further, payload restrictions mean that the use of inertial sensors to measure these disturbances is not feasible. Also, the quality of the images obtained from the miniature transmitting video camera is very poor and this greatly increases demands on the vision system. Finally, the large disturbances and poor image quality result in large pose uncertainty and the estimates must be correctly filtered before they can be used by a control system. This thesis shows how these issues can be addressed to enable the successful automatic flight of the MAV.

An automatically controlled MAV could be used in a large range of applications [Amidi *et al.* 1998], such as search and rescue, law enforcement and cinematography. Santana & Barata [2005] discuss the control of an unmanned helicopter for use in mine-clearing. MAVs can be used for automatic surveillance or mapping. For example, the mapping and monitoring of a rapidly changing environment such as a refugee camp could be accomplished by a miniature flying vehicle controlled by a visual system, without the need for a human operator. Currently, the only automatic systems for mapping camps use satellite images [Bjorgo 1999], which have limited resolution and, more importantly, are restricted on cloudy days. The automatic monitoring of difficult-to-get-to areas such as the undersides of bridges or the outsides of tower blocks would also be an ideal application. Recently, manually controlled blimps have been commercially employed to quickly and harmlessly inspect the inside of Cathedrals[1] and the visual control of blimps is discussed by Zhang & Ostrowski [1999].

[1]Skycell, `www.skycell.net`, last accessed Nov. 2005

1.1 Thesis Organisation

The visual tracking theory presented in this thesis is applied to the task of automatically controlling a four-rotor MAV, using the system described in Section 1.2. Chapter 2 introduces related work and Chapter 3 describes in detail the methodology of an existing visual tracking scheme, on which the new theory is based.

Chapter 4 presents a multi-modal extension to RANSAC which allows the tracking system to consider multiple pose hypotheses. The chapter opens with further discussion of the difficulties faced by a visual tracking algorithm operating on the images obtained from the MAV. To address these difficulties a model-based tracking scheme, which follows strong edge features, is used. Edges in image intensity are particularly stable features and hence provide high tolerance to the poor quality images. However, in order to tolerate large, unpredictable MAV motions, it is necessary to assume relatively little about the pose of the camera in one frame based on information from previous frames. As this prior knowledge is weakened, it becomes necessary to use the multi-modal extension to RANSAC to consider multiple possibilities for the pose of the camera at each frame. The resulting technique produces a particle-like representation of the multi-modal distribution, but is data driven and hence can operate efficiently with relatively weak prior estimates.

When operating in real-time however, evaluating the probability of many hypotheses is limited by computational constraints. Chapter 5 presents a solution to this by clustering measurements so as to reduce the complexity of high-dimensional parameter estimation problems. Clusters of measurements which are affected only by a subset of the parameter space are found. Each such cluster can then be used independently from all other measurements to isolate decisions about certain parameters. The method generates these clusters dynamically for each new video frame and hence, even though the 6D tracking problem cannot be partitioned in the traditional sense, achieves substantial computational reductions.

The tracking system presented is capable of providing reliable pose estimates in real-time. The remaining task is the automatic control the helicopter. To provide a suitably damped response, the controller must use estimates of the MAV's velocity, as well as its pose. Chapter 6 shows that obtaining sufficiently accurate estimates using conventional filtering techniques comes at the penalty of a slow response to the large disturbances suffered by the helicopter. Instead, a multiple hypothesis filter which employs a dynamic programming algorithm is presented. This filter is responsive, while yielding stable and accurate estimates of pose and velocity.

Finally, Chapter 7 describes the development of a controller which adjusts the attitude of the MAV from the pose and velocity estimates. The demands for the helicopter's attitude are in turn generated by a controller which controls its position. This nested controller is able to reliably stabilise the helicopter and results from test flights in two situations are presented.

Chapter 8 summarises the contributions of this thesis and suggests areas which require further investigation.

1.2 System Overview

The prototype MAV control system developed in this thesis is shown in Figure 1.1. The MAV used is a small, battery powered, consumer model helicopter (see Table 1.1), with its supplied radio control transmitter. A four-rotor device was chosen since these are significantly easier to control than conventional single-rotor helicopters. This can be easily seen when performing manual flights; also in the literature the control of single-rotor helicopters (even when accurate pose and velocity estimates are available) is an active area of research in its own right [e.g. La Civita *et al.* 2002] whereas the control of four-rotor helicopters is typically described along with other research [Altuğ 2003; Chen & Huzmezan 2003]. It is, of course, possible to fly the helicopter by hand; the author was able to hover the helicopter with reasonable accuracy after approximately 50 hours of practice.

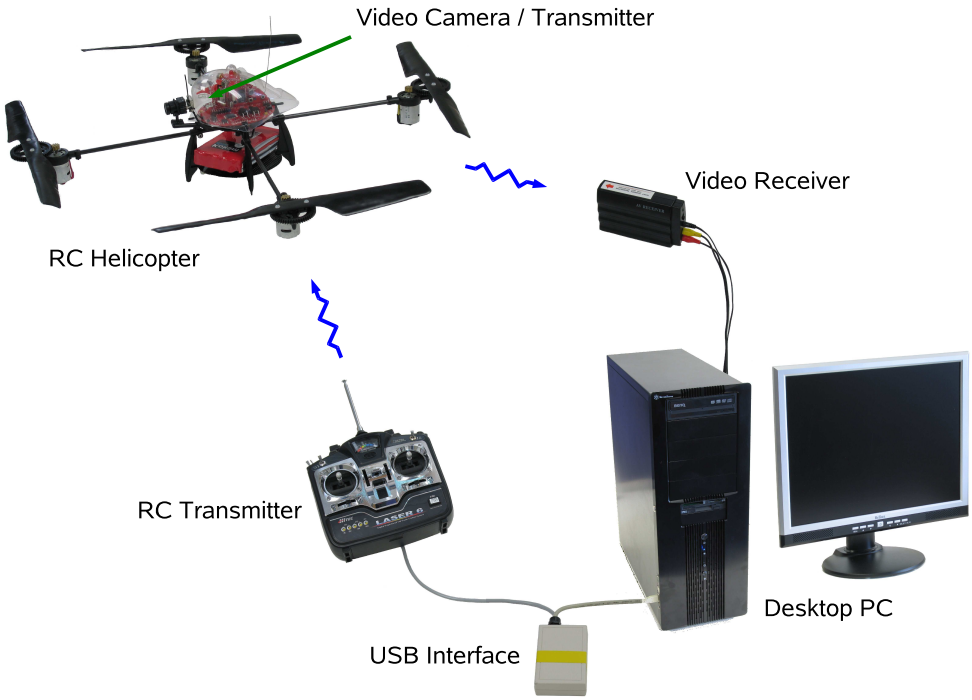


Figure 1.1: Overview of the complete closed-loop helicopter control system.

No. of Rotors	4
Rotor Diameter	290mm
Overall Diameter	750mm
Weight excluding Battery	386 grams
Weight including Battery	541 grams
Maximum Payload	12 grams
Flight Time	12 minutes
Radio Control	4 channel
Camera Transmission	2.4GHz, 10mW
Camera Imaging	365K pixels, 380 TV lines

Table 1.1: Model helicopter specification.

A single stream of images from the helicopter is captured by a sub-miniature video camera with an integral radio transmitter. The camera weighs only 9 grams, but has a number of limitations; most importantly shutter speed and gain are automatically controlled. Particularly indoors, this results in significantly blurred or saturated images. The camera is equipped with a wide angle, manual focus lens. The manual focus (and large fixed aperture) results in defocusing as the depth of the scene varies but this effect is small compared to the optical resolution and blur. The wide angle lens is necessary to capture an image with the greatest depth range possible. This is vitally important when trying to obtain the camera pose using a monocular camera system. The wide angle lens introduces considerable radial distortion but this effect can be removed using the calibration techniques described in Section 3.2.2. A typical image from the camera is shown in Figure 1.2.

Figure 1.2: A typical image obtained from the helicopter's onboard camera.

The images from the camera are sent by a 2.4GHz wireless link to a receiver. This produces a PAL video stream which is captured by a CX88-based capture card in a desktop computer. The images are captured as video fields rather than complete frames, since this significantly reduces the system time lag; a new image is obtained every 20ms (50fps) rather than every 40ms (25fps). The images are captured at 768x288 pixels, although the effective resolution of the camera is significantly less than this. Unfortunately the camera and transmission link produce images which are constantly subject to considerable pixel noise,

measured as having a standard deviation $\sigma > 5$ with intensities in the range $0 - 255$. Further, the transmission link often introduces significant errors in the image. Occasionally these are in the form of missed or mangled frames as shown in Figure 1.3. More commonly, the image is intact but parts contain errors as shown in Figure 1.4. These errors are typically caused when the transmission path is obstructed, when the motor current demands change rapidly or when the helicopter moves abruptly (e.g. when it bumps into the floor). Further difficulty is caused by these errors typically lasting for several consecutive frames.

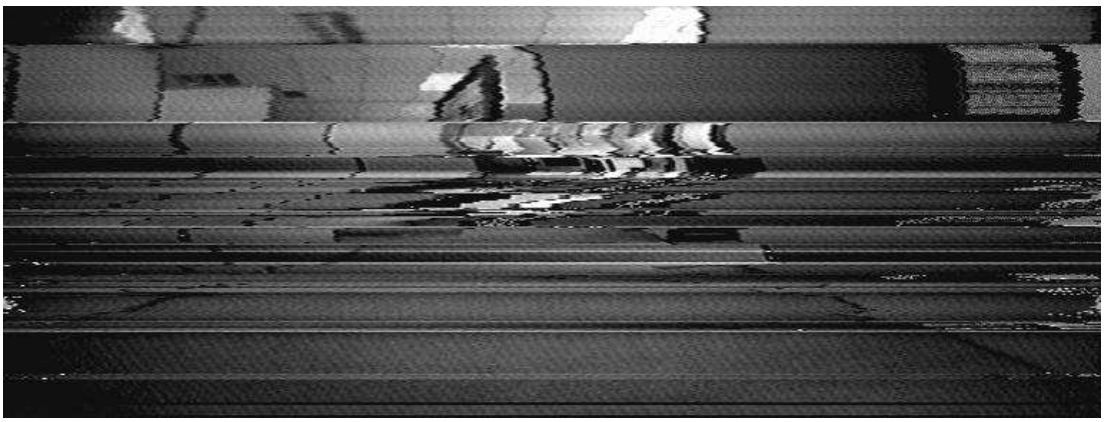


Figure 1.3: Occasional frames are severely mangled by the transmission system.

A desktop computer is used to execute the algorithms described in the following chapters. Specifications for the computer are given in Table 1.2 (and this was throughout this thesis unless otherwise stated). Most algorithms were implemented in C++ using the TooN numerics library[1] and CVD vision library[2]. The complete helicopter control system is computationally expensive and to operate in real-time, a multi-threaded architecture was used to split the load between the computer's dual processors (see Section 7.1).

The visual tracking algorithm described in Chapter 5 gives an estimate of the helicopter's pose. This is filtered as decribed in Chapter 6 and passed to the control system described in Chapter 7, which computes the required helicopter motor speeds. A custom made USB interface (Appendix B) is used to transfer

[1] `http://savannah.nongnu.org/projects/toon`
[2] `http://savannah.nongnu.org/projects/libcvd`

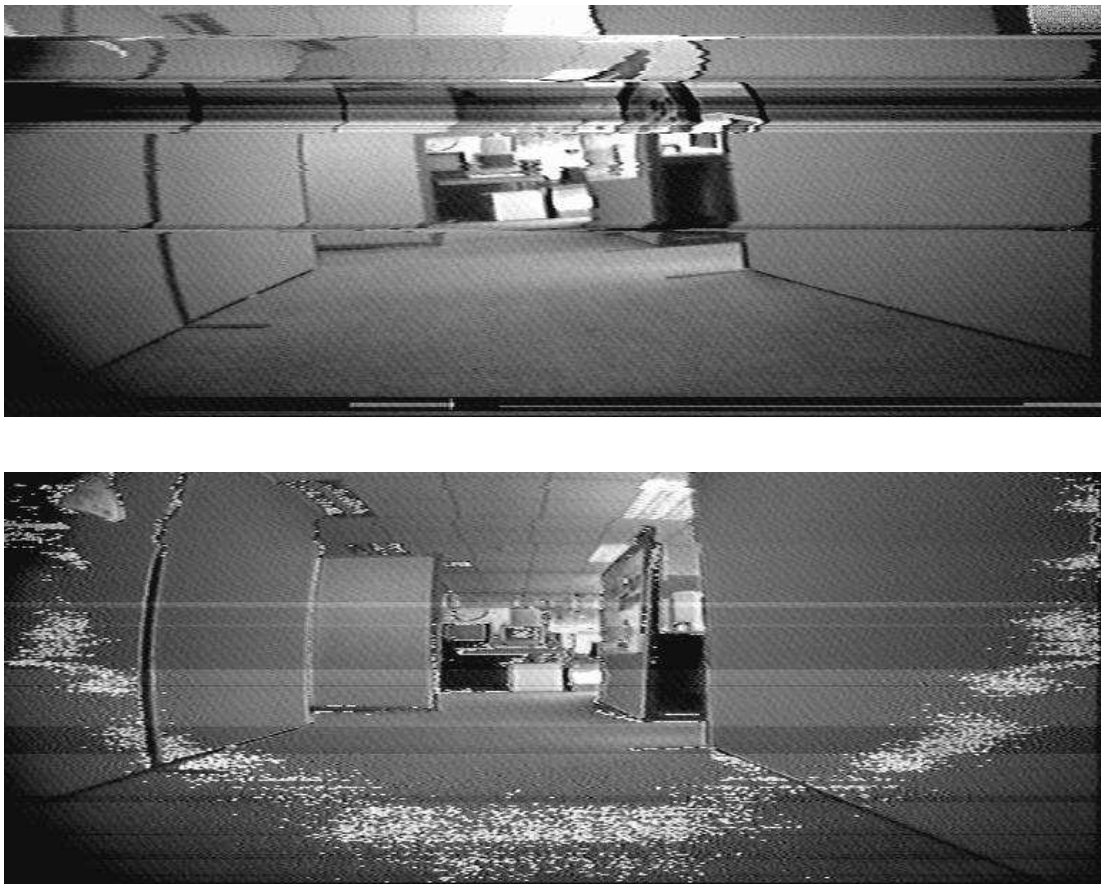


Figure 1.4: Often, frames from the camera contain smaller errors, but these can typically be tolerated by an edge-based tracking system.

No. of Processors	2
Processor Type	Opteron 252
Processor Speed	2.6GHz
Memory	2GB
Graphics Card	GeForce 6800GT (PCI Express) 256MB
Operating System	GNU / Linux (2.6.12)

Table 1.2: Desktop computer specification.

these demands to a standard radio control transmitter, which in turn transmits them back to the helicopter.

1.3 Publications

Much of the novel theory described in this thesis has been peer reviewed and presented at conferences:

Kemp & Drummond [2004] describes the multi-modal extension to RANSAC, which is here described in Chapter 4. An oral presentation of the work was given at the British Machine Vision Conference and the paper was awarded the 'BMVA Best Science Paper' prize. An extended version of the paper is expected to appear in the BMVC special edition of the journal 'Image and Vision Computing'.

Kemp & Drummond [2005] describes the dynamic measurement clustering technique, details of which are given here in Chapter 5. It was accepted for oral presentation at the International Conference on Computer Vision.

A journal paper combining the clustering technique with the filtering technique described in Chapter 6 and results showing successful flights of the helicopter has been prepared and submitted to a combined issue of the International Journal of Computer Vision and the International Journal of Robotics Research on 'Vision and Robotics'.

Chapter 2

Related Work

This chapter presents a review of previous work relating to the visual guideance of MAVs, starting with a discussion of current applications. Previous work in the central field of visual tracking is summarised in Section 2.2. Often it is not possible to *uniquely* determine the camera pose in every video frame and tracking robustness can be increased by postponing decisions until more evidence is available. Previous ideas relating to this field of Multiple Hypothesis Tracking (MHT) are described in Section 2.3. As shown in Section 1.2, the images obtained from the MAV are poor, and image measurements will inevitably contain significant errors. Robust methods to estimate parameters, such as camera pose, in the presence of such errors are discussed in Section 2.4. Chapter 4 will include an improved method for detecting image edges and earlier work in this field is summarised in Section 2.5. Section 2.6 describes existing techniques for filtering the tracking results, both to reduce noise and to provide a prior prediction for future frames. Section 2.7 then puts the novel ideas which are presented in this thesis into the context of these related works. Section 2.8 concludes with a review of previous work combining vision with MAVs.

2.1 Applications of Visually Guided Robotics

Applications of robotic machinery are both widespread and diverse. Often robots are used to replace human operators as their speed, reliability, precision and repeatability make them ideal for manufacturing tasks. Equally they may be used in locations or situations where it is unsafe or impractical for humans to work. Generally, it is essential for a robot to locate itself relative to surrounding objects. In the case of a fixed robot arm, the system must know the position of the components on which it is to work. In the field of mobile robotics, a means of locating the robot in the world is required. Sometimes, this can be achieved by limiting the robot to function only in an *instrumented* area - where devices (e.g. coloured lines or ultrasonic beacons) are added at fixed locations in the working environment. However, it is often preferable for robots to be able to operate in non-instrumented environments and this is typically achieved using a vision system.

The topic of visually guided robotics covers any robot system which is controlled using information obtained from a visual imaging system. Currently, such technology is mainly used to aid fixed manufacturing robots. Often in production systems parts arrive in uncontrolled orientations and positions and must be accurately located before automatic operations can proceed [Berger *et al.* 2000]. While other technologies can be used to locate parts, the simplicity and speed of a visual system make it an attractive choice. This is especially the case if a vision system can simultaneously be used to inspect the raw or manufactured parts.

In addition to applications for fixed robotics, another important domain for visual guidance is that of mobile robotics. Current applications are generally restricted to land based systems, for example robotic transport devices. Automatic transport systems are proven to work well in instrumented environments - besides robotic pallet trucks, unmanned train networks are now common in airports, or for example the Docklands Light Railway in London. Mobile robots can also be used in difficult or dangerous situations. The inspection of critical parts of nuclear reactors can be safely accomplished using a mobile robot equipped with a

video camera and other specialist sensors [Kita *et al.* 1999]. While such robots are currently guided by humans, a system which could guide the robot automatically would significantly help with accuracy and the avoidance of human error. Alternatively, current systems could be augmented with operator aids. For example, visual information could be used to increase control stability so that humans can drive the robot with greater ease, or third-person perspective views of the robot in its environment could be generated to aid visualisation.

2.2 Real-Time Visual Tracking

A large class of research explores the tracking of objects which move before a camera. Typically tracking is performed by locating distinctive image features, which are ideally both uniquely identifiable and invariant to pose and lighting changes. Features used include histograms of pixel intensities and colours [Comaniciu *et al.* 2000], planar textures undergoing affine deformations [Jurie & Dhome 2002] and exemplar-based contour matching [Toyama & Blake 2001]. Such techniques could be used to determine camera pose by triangulation if several such static objects are tracked. However, greater reliability in tracking pose is achieved by using many smaller features, which gives the opportunity to reject outlier measurements.

Existing pose tracking systems can be split into two groups: those which rely on a predetermined model of the tracked scene and those which learn the surroundings as the sequence progresses. Encouraging progress has been made recently in the latter group [Davison 2003; Se *et al.* 2001]. Since the manual creation of models is time consuming, such systems which automatically create or adapt models are very beneficial. However, where the surroundings are expected to be reasonably static, exploiting the prior knowledge contained in a model will aid tracking performance. In particular, model-based systems can more easily reject unmodeled features and so are more tolerant of non-static elements in the images, such as passing humans.

A common model-based tracking technique is to match point-like image features. Often, point features are matched from frame to frame [Simon *et al.* 2000] but such sequential updating can cause drift in the feature templates, which leads to drift in the tracked pose. This can be eliminated by combining inter-frame matching with precomputed keyframes [Vacchetti *et al.* 2003], a learnt point model [Gordon & Lowe 2004] or an edge model [Rosten & Drummond 2005; Vacchetti *et al.* 2004]. While point tracking is applicable to a wide range of problems, it is troublesome for the MAV application described in this thesis. The nature of the helicopter's operating environment, together with the substantial motion blur and image noise produced by the sub-miniature camera, mean that point features are very difficult to extract or match at all but the coarsest of scales.

2.2.1 Early Edge-Based Tracking

An early method employing image edges [Cipolla & Blake 1992] uses 'B-spline' snakes to track apparent contours. At each frame, image intensity gradients on either side of sample points positioned along the snake are examined. Each sample point is effectively attached with an elastic membrane to the point of largest intensity discontinuity and the snake is hence pulled towards the nearest contrast edge. Often however, models of the scene can be used to impose large constraints on the image features and thus increase robustness significantly. One of the earliest real-time, marker-less, model-based 3D visual tracking systems is RAPiD [Harris & Stennett 1990]. The aim of RAPiD is to keep an accurate estimate of the six system state parameters which describe the pose of the camera relative to the known 3D object. At each new video image the pose estimate is updated according to a motion model which predicts the motion based on recent measurements. Measurements of the video image are then made to correct this estimate to that observed in the image.

The predicted pose is used to render a 3D model which gives the expected location of edges in the image. A one-dimensional search for an image edge is then made in the local vicinity of various sample points, which are located along the

predicted edges. This local search at a small number of points (typically 20-30 per frame) is the main factor contributing to the computational efficiency of the method. Having obtained an error measurement for each sample point, the six pose parameters are updated to minimise these errors. A Jacobian matrix describing the differential of each sample point error with respect to each of the pose parameters is calculated and this linearisation is used to find a least-squares solution.

Harris and Stennets' RAPiD system was demonstrated to work at a full frame rate of 50Hz, although at the time this required precomputation of the visibility of each sample point from different camera positions. Advances in processing have led to many systems which are based on the ideas used in RAPiD. However, at the time, alternative approaches were also presented. Both Lowe [1992] and Gennery [1992] use specialist hardware to perform entire image processing at speeds approaching real-time. Lowe uses a Marr-Hildreth edge detector to obtain all the edges in the image. A tree style search is then used to match the model edges to those found in the image. At each node, representing a possible pose, the search is guided by including or removing image edges based on the probability that the match is correct for that pose. This probability is calculated based on the perpendicular edge distance, relative orientation and lengths and the predicted measurement covariance. The pose parameters, given a candidate set of matches, are then calculated using a least squares solution. Although the computational complexity of this scheme meant that the frame rate was only 3-5Hz, the method allowed articulated objects to be tracked and is robust to background clutter and significant occlusion.

2.2.2 Robust Tracking Schemes

Several improvements to the robustness of RAPiD are presented by Armstrong & Zisserman [1995]. The main concern is with the identification and elimination of incorrect edges. Since measurements made from control points situated

on one model edge will not be independent (as assumed by Harris and Stennet), RANSAC is used to find sets of measurements which agree with the specific primitive for that edge (e.g. straight line or conic). If the measurements are not sufficiently consistent then that edge is not used for that update. Using the remaining edges the pose estimate is updated but rather than a simple least squares solution, case deletion is used to identify and remove occasional incorrect edges (which are self-consistent but not consistent with the whole model). In addition to these measures to aid robustness the authors also note that the reliability of individual edges may vary over time (for example due to position or lighting changes). A decaying average of the frequency with which each primitive is correctly found is used to weight the confidence attached to it.

Such removal of outliers is very important in a scheme which uses a least squares solution because large errors due to incorrect matches will have a significant effect on the solution. Instead of actually removing outliers, an alternative is to replace least squares with a robust M-estimator. This reduces the effect of outliers by minimising a function (other than the least squares norm) which more accurately models the expected error distribution. This effectively allows for a higher probability of large errors caused by such incorrect matches. More details on robust estimation are described in Section 3.2.4.

An M-estimator approach is used as part of the system presented by Marchand *et al.* [1999]. Here a velocity model (such as a Kalman filter) is not required. Instead, a large-scale 2D affine motion is first determined by performing perpendicular edge searches. The affine transformation is calculated using a robust M-estimator. The full 6D pose is then refined in an iterative scheme to improve the matches between the model edges and areas of high intensity gradient in the image.

Simon & Berger [1998] use both a robust estimator and outlier rejection. Again a velocity model is not used but instead the location of the object is predicted using the normal optical flow measured in the image. An active contour (snake) is initialised at the predicted location and used to find the nearest image edge.

Having found candidate image edges a two stage approach is used to compute the pose. First at a local scale a robust estimator (either an M-estimator or Least Trimmed Squares) is used to adjust the residual distance from each model feature point to the image edge snake. Feature points with a large residual are then assumed to be outliers (having had little effect on the robust estimator result) and are removed. Secondly, at a global scale, an M-estimator is again used to adjust the pose parameters to best fit all the feature residuals. The robustness at this stage allows entire features which are accurately localised but incorrectly matched to be removed. The system also allows for model features to be added or removed from the currently tracked features as they move on or off the image. Although the system is not implemented in real-time, tracking is successful with low quality images and copes well when the active contours are attracted to incorrect local minima.

Klein & Drummond [2002] improve the robustness of an edge based tracking system using information available from inertial sensors, which offer accurate indications of sudden motions but tend to drift over time. 'Sensor fusion' is used to offer two advantages to the tracking system. Firstly the information from the inertial sensors can be used to give a good prior prediction of the pose at each frame. This allows the system to tolerate rapid motions, whilst the vision system compensates for (and calibrates) drift in the inertial readings. Secondly, with fast rotations the captured image tends to become quite blurred. The simple edge detector used in most edge based tracking systems will eventually fail to find the blurred edges produced by such motions. However, inertial information can be used to predict the direction and width of the blur and the edge detector can be tailored to specifically respond to edges blurred in this way. Hence the system can track in the presence of large motion blur.

2.2.3 Tracking versus Initialisation

The field of 'position initialisation' is concerned with locating (often in six degrees of freedom) an object in a single image of a scene containing that object. In

contrast to tracking methods no prior assumptions of the object location are made. Although typically slower than tracking methods, pose estimation methods share a similar aim and provide the ultimate robustness to unpredictable motions.

The huge amount of data present in an image means that object and pose recognition methods must generally first reduce the image to a set of features such as lines, corners or pixel intensities around such features. Such features must provide excellent information compression by greatly reducing the amount of data whilst preserving the information necessary to discriminate between and locate objects. Much of the literature in the field assumes that these features can be accurately matched to features on a known library representation of the object. Given such correspondences, which are substantially correct, the problem is then to find an iterative scheme which will converge quickly to give the correct pose. Suitable error functions which can be minimised to find the correct pose are given by Wunsch & Hirzinger [1996] or Phong & Horaud [1995]. Lu *et al.* [2000] present a globally convergent scheme, whilst Rosin [1999] and Wunsch & Hirzinger [1996] use steps to increase robustness against incorrect correspondences. Phong & Horaud [1995] show that in the case of straight line features, the optimisation to find the six parameters which describe an object's pose can be separated into a search over the rotation parameters followed by a search over the translation parameters.

The task of selecting and matching suitable image features has also been well considered. A recent review and evaluation is given by Mikolajczyk & Schmid [2005]. Image feature types are generally chosen for their invariance to viewpoint and lighting changes. For example, Lowe [1999] describes a very popular method for giving scale invariance to local intensity patches around point features.

2.3 Multiple Hypothesis Tracking

One of the most robust tracking schemes to date is CONDENSATION [Isard & Blake 1998]. Rather than relying on a single-hypothesis robust estimator, the scheme

uses a multi-hypothesis approach. A representation of a multi-modal probability distribution for the model parameters is stored using factored sampling (effectively N point measurements from the distribution). At each new frame N new sample points are selected randomly from the old set (using replacement), with more probable points having a higher probability of selection. The new points undergo drift motion predicted by a dynamical model and then random motion to model process noise. Observations (measurements) in the image are made at each of the new sample points to determine the likelihood of each hypothesis. The contour model used is a parameterised spline, the initial shape of which, along with the motion model used in the drift stage, is obtained from a training set. The system is capable of real-time operation and is robust to large amounts of clutter due to the multi-hypothesis approach. Although the local minima at any frame may be incorrect if the tracker locks onto the clutter, the true object is often being correctly tracked by an alternative hypothesis and hence the tracker can later recover.

Several extensions or modifications have since been made to the CONDENSATION scheme (also known as a particle filtering approach [Gordon *et al.* 1993]). Deutscher *et al.* [1999] apply particle filtering to the tracking of a simple model of an arm. The authors observe that a traditional Kalman filter approach is not able to cope satisfactorily with singularities and discontinuities in the parameter space. For a human arm this is most obvious where the arm is stretched out to the side and is able to twist with virtually no observable change - yet to get from certain poses to others it must be tracked successfully through this singularity. The authors observe that the random sampling stage of particle filtering is able to cope well with this situation. Singularities are also a problem for the tracking of rigid scenes. With a simple model of the surroundings it is not uncommon for tracked features to 'line up' so that at least one of the pose dimensions cannot be accurately determined.

Probably the biggest difficulty with the particle filtering approach is the large number of sample points required to adequately model multi-modal hypotheses in higher-dimensional spaces. Deutscher *et al.* [2000] wish to track an entire ar-

ticulated human body model with 29 degrees of freedom (dof). Despite relatively clean images with no background clutter this is a difficult task and even with 40000 particles (taking over 30 hours to process four seconds of video) the particle filter approach fails to track successfully. The authors propose a solution they call Annealed Particle Filtering. Instead of modelling the probability density at each frame using factored sampling, only the single mode or mean pose is stored. This enables the use of a simpler weighting function instead of having to evaluate the likelihood of each particle. An annealing approach is used to find the single highest peak from the multi-modal weighting function response. The simulated annealing procedure is effectively a coarse to fine search, initially using a very broad weighting function to coarsely smooth the response and then to gradually home in on the largest peak. Whilst the resulting system is too slow for real-time operation, a multi-layer annealed search is able to track their test sequence over ten times faster than basic CONDENSATION.

The annealed particle filter is an interesting contrast to the work of Deutscher *et al.* [1999]; annealing does not propagate the multiple hypotheses from one frame to the next, yet it is able to track past the singularities discussed by Deutscher *et al.* [1999]. Problems with singularities are presumably avoided by the annealing approach allowing a broad range of the parameter space to be searched at each time step. Hence even if the most likely pose obtained at a singular point is incorrect, the search at the next frame will be wide enough to find the correct pose.

Layered sampling, a similar coarse-to-fine approach, was introduced by Sullivan *et al.* [1999]. With standard particle filtering, if the likelihood function has a narrow peak relative to the width of the prior distribution, many particles are needed to ensure that one lands within the likelihood peak. If coarse versions of the likelihood function are available, this problem can be avoided by filtering and resampling particles at successively finer scales.

An alternative method of reducing the computational load of particle filtering is to use partitioned sampling. This idea, which was introduced by MacCormick

& Blake [1999] and applied with more detail to hand tracking in MacCormick & Isard [2000], uses a divide and conquer approach. Rather than attempting to 'search' the entire configuration space (by allowing the particle filtering drift stage to cover the space), the search is performed first in one set of dimensions and then in the next. Clearly this is only possible when the configuration space can be split such that the dynamics used in the second stage do not affect the projections of the particle positions into the first stage. However, for articulated models this will often be a reasonable assumption and the system offers good performance improvements.

While MacCormick's work uses manually defined partitions, a particle annealing scheme presented by Deutscher *et al.* [2001] automatically exploits some of the benefits, without needing explicit partitions. Hierarchical partitions of the search space are automatically formed when all measurements agree on parameter values. A crossover term is also employed to allow the search space to split into non-overlapping parallel partitions. However, this method suffers from the limitation that it requires common parameter values be estimated from *all* of the measurement data, which makes it difficult to use in the presence of large motion disturbances or significant clutter.

The approach used in particle filtering methods is to store the current set of hypotheses as point samples on a continuous probability distribution. Although this has many advantages, it is worth mentioning an alternative approach which uses a discrete set of hypotheses, each with its own error distribution. An efficient means of implementing a classic Multiple Hypothesis Tracking (MHT) algorithm is given by Cox & Hingorani [1996]. A matrix representation is used to show the assignment between measurements and geometric features and is augmented to include columns representing spurious and new measurements, so that valid hypotheses have only a single non-zero in any row or column. In brute force MHT all possible hypotheses are enumerated at each frame and for each the likelihood is evaluated. The efficiency introduced by Cox and Hingorani comes from being able to find the k-best hypotheses directly, in $O(k^2)$ time, by recasting the problem as a weighted bipartite matching problem in which nodes on one side

are the measurements, nodes on the other are the features and the edges are the log likelihood probabilities. This system, combined with a pruning scheme and a probability based on the Mahalanobis distance and cross correlation test is shown to give encouraging results with reasonable computation times. However the authors note that the introduction of cross correlation based matching was necessary to make the matching uncertainty manageable but also note that such matching is prone to errors in some circumstances.

Cham & Rehg [1999] apply a multiple hypothesis approach to the tracking of an articulated human figure. Rather than proposing possible hypotheses and then evaluating their probability, the modes of the probability distribution are tracked directly. This could be performed using a particle filter representation of the state but the high number of degrees of freedom makes this difficult. Instead the multiple hypothesis distribution is modelled as a piecewise combination of many Gaussian distributions. A method for producing samples from this distribution is presented and the state probability is measured from the image to redefine the distribution for the next time step. Encouraging results are shown with the successful tracking of a 38 dof human model over a dance sequence. A fully Bayesian approach is applied by Ringer & Lasenby [2002] to MHT of human figures by employing distinctive markers on the human to remove any matching ambiguity and hence to remove the need for particle filtering methods.

Work which uses multiple hypotheses to achieve both localisation and tracking has been presented by Arras *et al.* [2002] who assume that features of the environment (e.g. lines or objects such as fire extinguishers) can be accurately extracted and recognised but that spatial ambiguity remains (e.g. a door can be accurately identified as a door but not a particular door). The system proposed uses a search tree to return hypothesis sets of observation-to-model matches which fulfil a variety of geometric constraints. The multiple hypotheses are split or pruned as appropriate when more features are observed. The resulting system is demonstrated using only infinite line features and is shown to work well, including tolerance to the robot being suddenly moved (kidnapped) and it colliding with obstacles. However, processing is substantially slower than real-time and it is not

clear how well the system would scale to more than the three degrees of freedom currently tracked.

A multiple hypothesis approach is applied to the tracking of a hand by Stenger *et al.* [2003]. Hypotheses are represented as a Bayesian posterior probability but rather than using a particle filtering approach, a tree-based filter is employed. Nodes at each level of the tree represent non-overlapping areas of the state space, at increasingly fine detail. The tree provides an efficient method of concentrating computation on significant regions of the state space, since branches of more likely nodes can be explored first, thereby increasing the density of posterior probability samples in likely regions.

2.4 Robust Estimation

The Random Sample Consensus paradigm (RANSAC) was originally proposed by Fischler & Bolles [1981]. Though much of that work was devoted to the 'location determination problem' which is not relevant here, the RANSAC method has been used widely throughout computer vision and particularly for epipolar geometry constraints and 3D motion estimation (for a review see Torr & Murray [1997]). RANSAC is used to estimate parameters from a set of measurements and is truly robust to outliers in the data. A minimal set of measurements to constrain the model is randomly chosen and the remaining data points are tested to see if they agree with this hypothesis (the consensus test). The procedure is repeated several times and the hypothesis which has the largest number of consensus points is chosen. A final 'classic' optimisation can optionally be performed using only the consensus points. The authors observe that the random sampling stage may be guided by a prior probability if available.

Despite the success of RANSAC, it is clear that since many hypothesis sets must be considered (depending on the expected proportion of outliers), it is not particularly fast. Several different improvements, aiding both speed and accuracy, have since been proposed.

Randomised RANSAC was proposed by Chum & Matas [2002] to increase the speed of hypothesis testing. The idea is simple - an initial consensus test is performed in which a small subset of the remaining data points is first tested. The authors propose a very simple initial check where d data points are randomly selected. Only if all d points are within the consensus range the entire data set is checked for the exact consensus value. Despite the simplicity of this test, the approach offers a considerable speed improvement.

A Bayesian approach to RANSAC is presented as part of work for long range feature matching by Cham & Cipolla [1998]. The prior probability distribution of model parameters is utilised twice. Firstly, as suggested by Fischler and Bolles, prior probability is used to weight features during the random selection stage. Hence features which are more likely to be correctly matched are selected with a greater frequency. Secondly, the prior probability is used when calculating the model parameters from the sparse data set to find the maximum *a posteriori* (MAP) estimate.

Torr & Zisserman [2000] suggest that, rather than using the simple consensus count to score each random sample, a more meaningful measure should be used. The authors note that a magnitude limited error value can be used at no additional computation cost and that in general this improves results. If additional computation can be tolerated then the log likelihood can be calculated instead. The correctly matched measurements are assumed to have a Gaussian error distribution, whilst incorrect matches are assumed to be drawn from a uniform distribution. However a mixing parameter which determines the proportion of correct matches for this statistical model must be obtained. In contrast to Cham & Cipolla [1998], no prior information is made available, so instead the parameter is found using an iterative process which maximises the log likelihood for that sample set. The resulting scheme is shown to offer superior results to RANSAC.

Whilst maximisation of the log likelihood function is a logical extension of RANSAC, the iterative scheme for finding the mixture parameter is rather dubious since this probability indicates prior knowledge and should not depend on the hypothesis.

This issue is addressed by Tordoff & Murray [2002] where it is shown that, for a good set of real images, there is little to be gained by estimating the mixture parameter and that in practice it can be set as a constant. However, there is often information available as to the quality of each match. In this case the correlation score for each feature is used to indicate the prior probability that the feature is correctly matched. As well as being used to provide the mixture parameter for the log likelihood estimation, this prior is used to guide the random sampling (as Cham & Cipolla [1998]). A separate issue addressed by Tordoff & Murray [2002] is that temporal information can be propagated in multiple frame motion tracking: a prior probability of each feature being from the foreground, background or a mismatch can be obtained from its result in the previous match.

As mentioned previously, an alternative to specifically marking measurements as inliers or outliers is to apply a robust M-estimator to all the measurements. These minimise a function of the residual errors which is less biased towards outliers than the L_2 norm. Such estimators have been applied to both pose estimation [Wunsch & Hirzinger 1996] and tracking [Drummond & Cipolla 1999; Marchand *et al.* 1999; Simon & Berger 1998]. Although these schemes avoid any explicit searching of correspondence sets, unfortunately iterative solutions must be used and these find local minima rather than necessarily finding the global minimum. Convergence on the global minimum is dependant on a good initial point (prior prediction) from which to iterate and, as shown in Section 4.1, a poor prediction can lead to tracking being lost.

Another alternative to L_2 optimisation is to minimise the median squared error and this technique is applied to robust visual pose estimation by Rosin [1999]. The method can find the global minimum with a given confidence level using a RANSAC-like algorithm but unlike most RANSAC approaches the technique has the advantage that the computational order is linear in the number of data points.

A final class of robust estimation techniques use statistical methods. Simulated annealing [Kirpatrick *et al.* 1983] makes connections between optimisation and statistical mechanics and progressively optimises using a 'cooling' process. Possi-

ble solutions are represented by particles in the parameter space and at each step the particles are given a small random displacement. The displacement is kept if it results in an improved solution but also there is a random chance that it is accepted, even if it makes the solution worse. This allows particles to occasionally jump out of local minima, but as the temperature is cooled the chance of bad displacements being accepted is reduced. Unlike iterative techniques such as M-estimation, simulated annealing is often able to avoid local minima. The gradual cooling also helps divide and conquer as particles are progressively concentrated on more probable areas of the parameter space.

2.5 Edge Detection

A significant number of visual tracking systems work by locating edges in each image. Although detailed research into edge detection methods has been conducted for the processing of entire 'static' images, the methods used in real-time tracking are normally simplistic and local. For example, CONDENSATION hypothesises the position of an edge and then examines the image only within a few pixels of that position, using intensity differences to determine the probability of there being an edge at that point.

A review of many traditional methods for finding image edges is given by Smith [1997]. A classic work on image edge detection is Canny [1986]: Image edges are modelled as step intensity changes with additive Gaussian noise across all readings. An optimal filter is derived which maximises the product of a detection probability criterion and a localisation accuracy criterion. It is then shown that this optimal filter can be well approximated by the differential of a Gaussian, which in turn is well approximated by the difference of two Gaussians. Canny observes that this filter is not applicable to finding boundaries between textures and that different sized filters must be used to find accurately image features of different sizes. A threshold value is used on the filtered image to determine initially whether each pixel is an edgel or not, and non-maximal suppression and

hysteresis are then used to help create unbroken chains of edgels - pixels which are believed to lie on an edge.

Shahrokni *et al.* [2004] discuss the need for finding *texture* edges or boundaries using local searches (rather than processing the entire image) in order to achieve real-time tracking operation. The work borrows hidden Markov random field techniques from the texture segmentation literature but shows how they can be applied to find the most probable texture change-point along a single scanline of pixel intensities. Under certain assumptions a closed form solution can be found which leads to a very efficient algorithm. More details on this technique, which is also used here, are given in Section 4.3.

2.6 Pose Filtering

Although many of the tracking schemes discussed effectively implement a filter to update the camera pose, filtering has also be studied in its own right. A filter has two distinct uses in a tracking system. Firstly, if the result of the tracking stage is noisy, a filter may be able to smooth the result by statistically modelling this noise. Secondly, all tracking systems require a prior prediction of the pose for a new video frame (see Section 3.1.1) and this must be provided by a filter or motion model.

The simplest form of pose filtering assumes that the pose measurement is correct and that the camera is stationary. Hence, for time $t + \Delta t$, the prior prediction of the system state $\hat{\boldsymbol{x}}_{t+\Delta t}$ is simply the posterior at time t, $\boldsymbol{x}_t$.[1] Such a model severely limits the speeds at which the camera can be moved. Instead, a 'constant velocity' model is often used:

$$\hat{\boldsymbol{x}}_{t+\Delta t} = \boldsymbol{x}_t + \Delta t \; \boldsymbol{v}_t \quad \text{and} \quad \boldsymbol{v}_t = \alpha \boldsymbol{v}_{t-\Delta t} + \beta(\boldsymbol{x}_t - \hat{\boldsymbol{x}}_t)/\Delta t \;\; . \tag{2.1}$$

[1] For clarity here it is assumed that the system state (the current pose) can be represented as a vector and hence bold notation is used. Often, a linearisation about the current pose is used, as described in Section 3.2.

The two constants (α and β) control an overall decay in the velocity prediction (for a strictly constant velocity model, $\alpha = 1$) and add damping which is vital to avoid the posterior oscillating about the correct pose.

The Kalman filter [Welch & Bishop 1995] describes the optimal method for parameter estimation in the case when the state update and measurement equations are linear and the measurement and process noises are additive, Gaussian and independent (white). It can therefore be used to implement a 'constant velocity' model which assumes that the camera velocity is constant apart from accelerations which are drawn from a Gaussian distribution.

For cases when the equations are not linear, the extended Kalman filter (EKF) [Welch & Bishop 1995] linearises the equations about the current state, for each update step. It has frequently been used with visual tracking, both as a means of smoothing the output and to provide a prediction for the pose in the following frame [e.g. Armstrong & Zisserman 1995]. The Unscented Kalman filter [Haykin 2001] is an alternative to the EKF and uses linearising approximations that are typically more accurate than the direct linearisations used in the EKF. However, the use of both these filters is still limited to situations where the process noise is largely Gaussian; as shown in Section 6.1.2 they do not work well when this is not the case.

For multi-modal distributions, it is possible to use a Gaussian mixture model to more accurately represent the distribution. This was applied to the tracking of human figures by Cham & Rehg [1999] in order to overcome ambiguities.

Particle filtering techniques [Arulampalam *et al.* 2002] can also be employed as a post-filtering stage for other tracking schemes, by including both pose and velocity in the state of each particle. As when used as part of the tracking system itself, particle filters work well even when the underlying process noise or measurement noise are not well approximated by a Gaussian distribution and when the likelihood distribution is multi-modal. However, particle filters are best used when the parameter space has low dimension and when the likelihood function can be evaluated cheaply. Unfortunately, tracking both pose and velocity

requires a twelve degree of freedom parameter space and reliable representation of the posterior requires a large number of particles.

The Unscented Particle filter [van der Merwe *et al.* 2000], applied to visual tracking by Rui & Che [2001], goes some way to reducing the number of particles needed to successfully represent the distribution by employing both the prior and the likelihood at the resampling stage.

2.7 Context of this Thesis

Visual tracking systems aim to follow the motion of image features, from which the pose of the camera can be obtained. Although a variety of features have been considered, the quality of images obtained from the MAV means that using edge features for which a 3D model is available gives the best chance of success. Edge based systems suffer from difficulties making accurate correspondences and the most robust schemes all consider multiple hypotheses or a multi-modal probability distribution. All such schemes inherently employ robust estimation techniques which allow for the presence of the outlier measurements that lead to the multiple hypotheses.

Unfortunately, the strict need for real-time operation produces a limit on the 'width' of hypotheses that can be considered and this directly limits the size of unpredictable motions which can be tracked by existing schemes. In this thesis, the idea is to use 'divide and conquer' techniques similar to those of MacCormick & Blake [1999] but to generate a posterior probability representation similar to those obtained by CONDENSATION [Isard & Blake 1998]. Using a data-driven approach should reduce the computation time required so that a greater number of hypotheses can be considered and robustness to unpredictable motions can be increased. Building on the automatic partitioning ideas of Deutscher *et al.* [2001], here automatic clustering of measurements will be used to quickly reject poor hypotheses and again allow a greater number of possibilities to be considered.

Robust visual tracking schemes are applicable to a variety of applications. Here, the specific task of controlling a miniature aerial vehicle is addressed and this requires improvements to existing pose filtering methods. Unimodal EKFs [Welch & Bishop 1995] can be used to provide accurate velocity measurements but respond poorly to non-Gaussian motions while particle filters [Arulampalam *et al.* 2002] introduce massive additional computational requirements. Instead, a Kalman filter 'mixture-model', rather like that used by Cham & Rehg [1999], will allow both accurate velocity measurements and the ability to mode-switch when necessary to follow sudden velocity changes more rapidly.

2.8 Miniature and Unmanned Aerial Vehicles

Miniature aerial vehicles are challenging to control for three reasons; they are under actuated, non-linear and subject to large disturbances. Probably the simplest devices to control are model blimps and the visual control of an indoor blimp is presented by Zhang & Ostrowski [1999]. Their main contribution is the avoidance of any explicit calculation of the blimp's pose. Instead, the control inputs are directly related to the motion of image features, with the inclusion of appropriate dynamic modelling. This is used to enable the blimp to follow the path of a coloured ball. A similar technique of visual servoing directly from image line features is described by Andreff *et al.* [2002].

Research at the University of Florida centres around small ($6-24$ inch wing span) aeroplanes. Vision has been successfully used [Ettinger *et al.* 2002] to provide roll and pitch estimates using the horizon line, from which reliable automatic flight has been demonstrated. Vision research continues to improve horizon detection [Todorovic *et al.* 2003] and is enabling useful flight 'missions' by providing image classification [Todorovic & Nechyba 2004].

The ability of helicopter based MAVs to take off vertically, hover and fly in reasonable air disturbances provides significant operational advantages over blimps

and planes. However, this comes at the expense of greatly increased modelling and control complexity, a detailed study of which is given by Prouty [1990].

The theoretical work by Shakernia *et al.* [1999] concentrates on the control dynamics of a large, single-rotor model helicopter, flown using a full GPS and inertial sensor suite. Vision is used to estimate the position and velocity of the helicopter relative to a planar landing pad. An analysis of the errors involved in such estimation and their effect on the control loop are given.

Sinopoli *et al.* [2001] also achieve control of a helicopter using GPS and inertial sensors. Coarse, laser-measured maps of the earth's surface are used to provide a global navigation plan. The authors propose the use of a visual system to refine and augment the coarse map of the terrain, in order to provide, in real-time, detailed route-points and perform obstacle avoidance. Given odometry information from the helicopter, optical flow is used to provide detailed depth maps and these are compared this with the coarse terrain map to obtain information about obstacles.

Amidi [1996] summarises work at Carnegie Mellon culminating in the successful test flights of a petrol helicopter (overall length 3.5m). Custom hardware is used to perform on-board processing of images from two cameras. Additional sensing is provided by differential GPS measurements and an onboard compass and gyroscopes, giving a total of 18Kg of extra on-board equipment. The controller used is a series of nested propotional-differential loops.

Dittrich [2002] investigates a methodology for the flexible integration of avionics systems with a large (overall length 3.6m) helicopter. The approach ranges from software to perform both software- and hardware-in-the-loop simulations to flexible hardware mountings. Analysis of sensing errors is described and results from real flights are given.

Buskey *et al.* [2003] describe recent work with a medium size (overall length 1.8m, 8kg total weight) single-rotor helicopter. Position sensing is achieved with an inertial unit (weighing 65g), differential GPS and a stereo vision system with

an onboard 733MHz processor. Control is achieved through a number of nested PI controllers. Heading, roll and pitch are controlled using PI control, where the integral term provides slow compensation for trim, main rotor speed and wind variations. A nested proportional controller calculates the roll and pitch demands from the user-provided velocity demand. It appears that no absolute positioning control is implemented. Both a stand-alone PID module and a nested PI controller are investigated for height control. From their graphs, the flights occur above the ground effect (the effect of the rotor down-draft when flying near to the ground).

Unfortunately, smaller single-rotor helicopters are relatively hard to control due to their light weight and the use of fly-bars (for a review see Shim *et al.* [1998]) but recently indoor, electric helicopters which employ four fixed rotors have become available. Chen & Huzmezan [2003] describe the identification of a non-linear model of such a helicopter and the design of a $\mathcal{H}_\infty$ controller. Results for real flights when attached to a two degree of freedom measurement rig are presented.

Altuğ [2003] presents work with a four rotor model helicopter similar to that used here. The pose of the MAV is determined using both an onboard camera and a separate floor-mounted camera with an automatic pan/tilt head. This, combined with coloured 'blob' markers on both the helicopter and the world, provides accurate and robust position estimates. Theoretical work relating to rotor placement and control loop design is also presented.

Chapter 3

Framework

This chapter details one method for visual tracking and its mathematical tools. This framework, which was described by Drummond & Cipolla [1999], provides the basis that Chapters 4 and 5 extend.

3.1 Edge-Based Visual Tracking

Model-based tracking is in essence very simple; as each new frame arrives, the pose of the model is adjusted so that it best matches the image. The pose is not obtained afresh every frame but is adjusted away from a *prior* prediction. This prediction may be the pose found for the previous frame or it may be based on a more complicated model of how the camera moves.

Tracking therefore has two important limitations. At the start of the process the camera pose must be initialised. As discussed in Section 2.2.3 this process may be automatic, but throughout this thesis the initialisation is performed manually. Secondly, if the camera motion is too large, or the tracking updates are incorrect, it is possible for track to be lost. Again, automatic methods to reinitialise in such circumstances could be used. However, these methods are not fast and the time lag introduced would be unacceptable when used for real-time control. Hence no

automatic procedure is investigated and the systems presented all assume that track is not lost.

3.1.1 Basic Tracking Algorithm

The following stages are used to process each new frame:

1. Frame Capture (Figure 3.1). With a real-time system, the process timing is controlled by the arrival of a new frame from the video capture card.

2. Pose Prediction. Using the frame time-stamp, the pose of the camera in the new frame is predicted using a motion model. This prior may simply be the pose obtained from the previous frame. Other models were discussed in Section 2.6.[1]

3. Model Render (Figure 3.2). The 3D model of the camera surroundings is rendered at the predicted pose. For an edge-based tracking system this generates a list of the expected locations (in the image) of straight edge segments.

4. Image Measurements (Figure 3.3). The video image is now analysed to locate the actual positions of the image edges. One method for performing this is described in Section 3.1.3.

5. Pose Update (Figure 3.4). The pose is adjusted away from the prior prediction so that the model rendered at the posterior pose best matches the edges actually found in the image. This mathematical process is described in Section 3.2.4.

[1] If frames arrive from the capture card at predictable intervals, it is possible to perform stages 2 and 3 before the new frame arrives. This does not affect the overall processing time, but reduces latency.

Figure 3.1: Tracking Stage 1: New frame arrives.

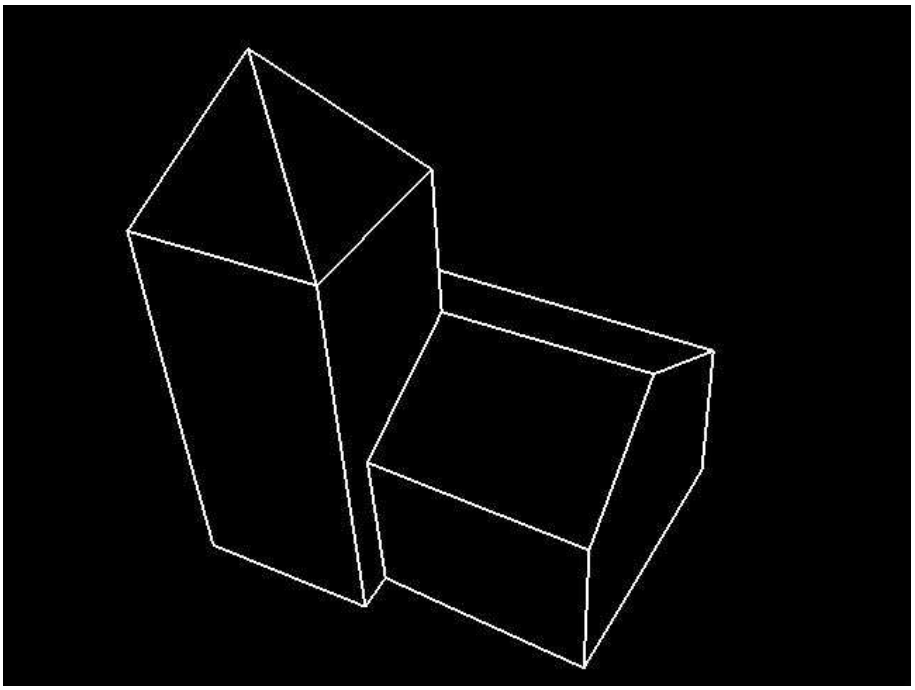


Figure 3.2: Tracking Stage 3: The 3D model is rendered at the predicted pose.

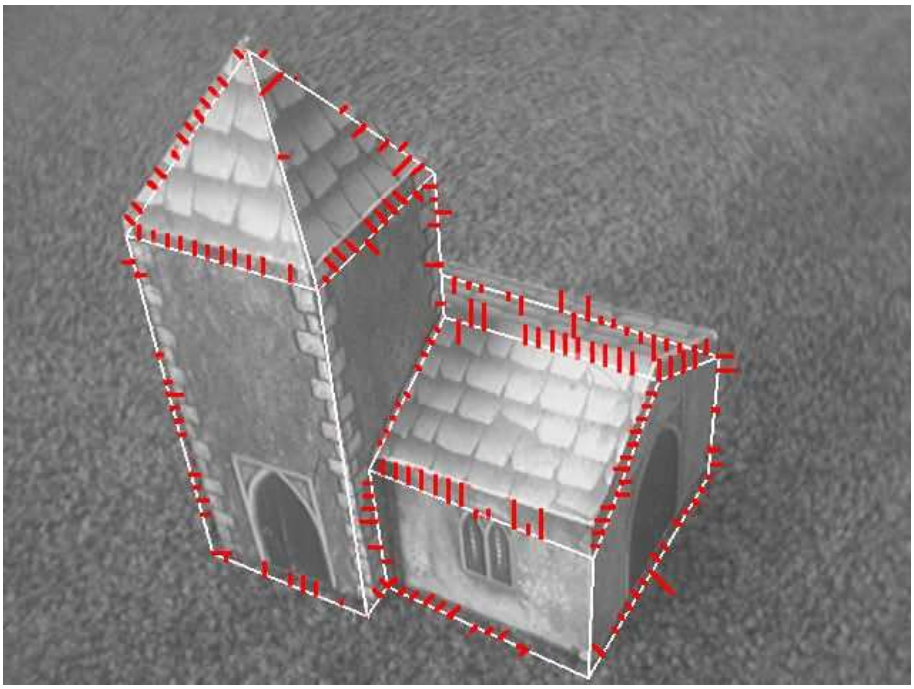


Figure 3.3: Tracking Stage 4: Edges near the rendered model lines are detected.

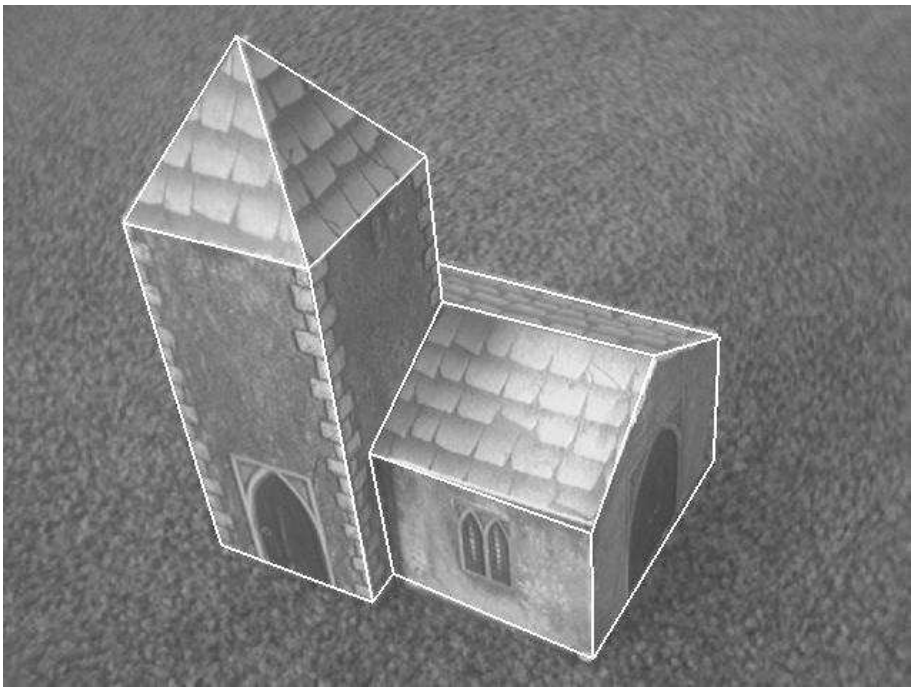


Figure 3.4: Tracking Stage 5: The pose prediction is adjusted to best align the model with the detected edges.

3.1.2 Model Rendering

A 3D model of the MAV's surroundings is constructed off-line. It contains the coordinates of the ends of straight edges in the world, such as a line between the floor and a wall or distinctive features such as picture frames or door surrounds. The rendering stage must determine the visibility of each edge or segment of it, and project the visible endpoints into the camera image. By including lists of edges which form closed faces in the 3D model file, a line-sweep algorithm can rapidly determine edge visibility. The result of the rendering stage is therefore a list of visible lines, each described by the image coordinates of its two endpoints.

Although a 3D model line may be visible in the sense that it is not obscured, it may not actually be visible or detectable in a camera image of the scene. There are two common examples of this: firstly when the line is between two areas of similar colour material and secondly when lines are too close together to be accurately distinguished. In many cases the robust measures contained in the tracking algorithm are able to tolerate such problems. However, for the system to scale to large environments, undetectable lines would need to be automatically culled. In a few occasions where this was necessary for the scenes presented in this thesis, manual limits on visibility were included.

3.1.3 Edge Detection

The starting point for edge detection is the image coordinates describing a straight line and the task is to find a nearby edge in the video image. Only the position of the image edge in the direction perpendicular to the model line is measured. This is because the 'aperture effect' means that detecting the position of the edge in the direction parallel to it will be impossible or difficult. The method given by Drummond & Cipolla [1999] starts by positioning *sample points* at regular intervals along the model line.

At each sample point a 1D edge search is performed, in both directions, perpendicular to the line (see Figure 3.3). In the simplest case, an edgel is detected where the difference between two consecutive pixel intensities exceeds a predetermined threshold value. To improve the noise tolerance and the directional response of the edge detector, the average of three neighbouring pixels in the direction parallel to the line can be used.

Walking through the image in the direction normal to the model line can be quite computationally expensive. Instead, a reasonable result can be obtained by rounding the actual search direction to the nearest of the 45° directions, since these can be searched easily. In this case it is of course necessary to adjust the distance measurement obtained back to the distance in the true line normal direction.

3.2 Mathematical Framework

3.2.1 Projection of a World point into the Camera image

Features in the model of the helicopter's surroundings have a 3D coordinate $(x_{\mathcal{W}}, y_{\mathcal{W}}, z_{\mathcal{W}})$ measured in a fixed 'world' coordinate frame. When viewed in a coordinate frame fixed with the helicopter's camera, the same point will have different coordinates, which depend on the pose of the camera. This transformation is easily represented if homogeneous coordinates are used for the model feature in the 'world' frame:

$$\boldsymbol{p}_{\mathcal{W}} \equiv \begin{bmatrix} x_{\mathcal{W}} \\ y_{\mathcal{W}} \\ z_{\mathcal{W}} \\ 1 \end{bmatrix} \quad . \tag{3.1}$$

Left-multiplication by a 4×4 Euclidean transformation matrix now yields the point in the 'camera' frame:

$$\boldsymbol{p}_{\mathcal{C}} = E_{\mathcal{CW}}\, \boldsymbol{p}_{\mathcal{W}} \ . \tag{3.2}$$

All Euclidean transformation matrices have the form:

$$E = \begin{bmatrix} & R & & \boldsymbol{t} \\ 0 & 0 & 0 & 1 \end{bmatrix} , \tag{3.3}$$

where R is a 3D rotation matrix ($|R| = 1$, $R^T R = I$) and $\boldsymbol{t}$ is a translation vector.

A model of the camera and lens is needed to find the projection of the point into the image frame. For a pin-hole camera, the image coordinates $\boldsymbol{u} = [u\ v]^T$ can be found as:

$$\begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 & 0 \\ 0 & f_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \boldsymbol{p}_{\mathcal{C}} , \tag{3.4}$$

where f_u and f_v describe the lens focal length in pixels and u_0 and v_0 give the location of the principal point in pixel coordinates.

3.2.2 Camera Distortion

Unfortunately, as shown in Figure 3.5, the wide-angle lens employed produces significant barrel distortion and so the linear model just described is inaccurate. Instead, it is necessary to use a polynomial model (see Zhang [2000] which also includes methods for calibrating the camera). This provides a relationship between the radius $\tilde{r}$ in the real, distorted imaging plane and the radius r in an undistorted version, given by:

$$r = \sqrt{\left(\frac{x_{\mathcal{C}}}{z_{\mathcal{C}}}\right)^2 + \left(\frac{x_{\mathcal{C}}}{z_{\mathcal{C}}}\right)^2} , \quad \text{where} \quad \boldsymbol{p}_{\mathcal{C}} = \begin{bmatrix} x_{\mathcal{C}} \\ y_{\mathcal{C}} \\ z_{\mathcal{C}} \\ 1 \end{bmatrix} . \tag{3.5}$$

The polynomial approximation takes the form:

$$r = \tilde{r} + \alpha_1 \tilde{r}^3 + \alpha_2 \tilde{r}^5 + ... , \tag{3.6}$$

but it is typically rearranged to:

$$\tilde{r} = r - \beta_1 r^3 - \beta_2 r^5 + ... \tag{3.7}$$

for computational efficiency. For the particular lens used, only the first three terms of this expression are used. The full camera projection including radial distortion is therefore given by:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{bmatrix} \begin{bmatrix} \lambda \frac{x_{\mathcal{C}}}{z_{\mathcal{C}}} \\ \lambda \frac{y_{\mathcal{C}}}{z_{\mathcal{C}}} \\ 1 \end{bmatrix} \quad , \quad \text{where} \quad \lambda = 1 - \beta_1 r^2 - \beta_2 r^4 \ . \tag{3.8}$$

Figure 3.5: A view of a calibration grid seen through the MAV camera, showing significant barrel distortion.

3.2.3 Camera Motion

As described in Section 3.1.1, tracking is concerned with the camera *motion* between consecutive frames. Such a motion will change the matrix $E_{\mathcal{C}\mathcal{W}}$ described above. Denoting $\mathcal{C}_i$ as the camera position in frame i, the Euclidean transformation matrix for frame 2 can be found from that for frame 1:

$$E_{\mathcal{C}_2\mathcal{W}} = E_{\mathcal{C}_2\mathcal{C}_1} E_{\mathcal{C}_1\mathcal{W}} \ . \tag{3.9}$$

The set of all possible E forms a representation of the 6-dimensional Lie Group SE(3), the group of rigid body transformations in $\mathbb{R}^3$. Such matrices have six

degrees of freedom and may be parametrised as a six-dimensional vector $\boldsymbol{\mu}$. The relationship is given by the exponential map:

$$E = \exp(\boldsymbol{\mu}) \equiv e^{\sum_{j=1}^{6} \mu_j G_j} \quad , \tag{3.10}$$

where G_j are the group generator matrices, which here take the values:

$$\begin{aligned} G_1 &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \; G_2 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_3 = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \\ G_4 &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \; G_5 = \begin{bmatrix} 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad G_6 = \begin{bmatrix} 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \tag{3.11}$$

Both the exponential and the logarithm ($\boldsymbol{\mu} = \ln(E)$) can be found in closed form: More information on the Lie Group SE(3) and its properties may be found in Tomlin [1997].

Although any Euclidean matrix can be parameterised as a six-vector $\boldsymbol{\mu}$, this representation is typically used when the matrix describes a small motion, as is the case for $E_{\mathcal{C}_2\mathcal{C}_1}$ in Equation 3.9. In order to solve for the motion, it is convenient to linearise the projection equations and obtain the partial derivatives of the motion of the feature in the image with respect to the six motion parameters. The use of the exponential map (Equation 3.10) makes this straightforward.

To find $\frac{\partial\{u,v\}}{\partial\mu_j}$ (where u and v are the new image coordinates in frame 2), it is

convenient to split the projection equations into smaller parts:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ 1 \end{bmatrix} \tag{3.12}$$

$$\boldsymbol{a} = \lambda \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \tag{3.13}$$

$$\boldsymbol{b} = \begin{bmatrix} \frac{x_{\mathbb{C}_2}}{z_{\mathbb{C}_2}} \\ \frac{y_{\mathbb{C}_2}}{z_{\mathbb{C}_2}} \end{bmatrix} \tag{3.14}$$

$$\boldsymbol{p}_{\mathbb{C}_2} = \exp(\boldsymbol{\mu})\, E_{\mathbb{C}_1\mathcal{W}}\, \boldsymbol{p}_{\mathcal{W}} \ . \tag{3.15}$$

The derivatives of each part can be easily calculated:

$$\frac{\partial\{u,v\}}{\partial \boldsymbol{a}} = \begin{bmatrix} f_u & 0 \\ 0 & f_v \end{bmatrix} \tag{3.16}$$

$$\frac{\partial \boldsymbol{a}}{\partial \boldsymbol{b}} = \lambda I + \boldsymbol{b}\frac{\partial \lambda}{\partial \boldsymbol{b}} \tag{3.17}$$

$$= \lambda I + \boldsymbol{b} \begin{bmatrix} -2\beta_1 b_1 - \beta_2(4b_1^3 + 2b_1 b_2^2) \\ -2\beta_1 b_2 - \beta_2(4b_2^3 + 2b_2 b_1^2) \end{bmatrix}^T$$

$$\frac{\partial \boldsymbol{b}}{\partial \boldsymbol{p}_{\mathbb{C}_2}} = \begin{bmatrix} \frac{1}{z_{\mathbb{C}}} & 0 & \frac{-x_{\mathbb{C}}}{z_{\mathbb{C}}^2} \\ 0 & \frac{1}{z_{\mathbb{C}}} & \frac{-y_{\mathbb{C}}}{z_{\mathbb{C}}^2} \end{bmatrix} \tag{3.18}$$

$$\left.\frac{\partial \boldsymbol{p}_{\mathbb{C}_2}}{\partial \mu_j}\right|_{\boldsymbol{\mu}=0} = G_j\ E_{\mathbb{C}_1\mathcal{W}}\, \boldsymbol{p}_{\mathcal{W}} \ . \tag{3.19}$$

The last follows since for small $\boldsymbol{\mu}$:

$$\frac{\partial \exp(\boldsymbol{\mu})}{\partial \mu_j} = G_j \ , \tag{3.20}$$

where G_j is a generator matrix as given in Equation 3.11.

Finally, for edge tracking where (u, v) is the position of a sample point and d is a 1D measurement of the image edge in the direction $\boldsymbol{n}$ (see Section 3.1.3), the partial derivative of this error measurement can be found as:

$$\frac{\partial d}{\partial \mu_j} = \boldsymbol{n} \cdot \frac{\partial [u\ v]^T}{\partial \mu_j} \ , \tag{3.21}$$

and these derivatives for all i measurements arranged into a Jacobian matrix:

$$J_{i,j} = \frac{\partial d_i}{\partial \mu_j} \ . \tag{3.22}$$

3.2.4 Tracking Updates using Least Squares

Having analysed the video frame to determine the location of image edges, the task is to adjust the camera pose to best align the modelled edges. Section 3.1.3 showed how a series of 1D measurements d_i could be obtained which described the perpendicular distance from each sample point i to the nearest image edge. Using the differentials described in the previous section, a camera pose motion described by $\boldsymbol{\mu}$ will result in a perpendicular motion $J_i^T \boldsymbol{\mu}$ of the ith sample point and hence ideally, $J_i^T \hat{\boldsymbol{\mu}} = d_i$. To simultaneously satisfy this equation for many measurements is not (in general) exactly possible and instead an error metric must be minimised. If the error measurements were independent and Gaussian, a least squares solution could be used:

$$\begin{aligned} \hat{\boldsymbol{\mu}} &= \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \ |J\boldsymbol{\mu} - \boldsymbol{d}|^2 && (3.23) \\ &= (J^T J)^{-1} J^T \boldsymbol{d} \ , && (3.24) \end{aligned}$$

where each row of J is the Jacobian of an individual sample point and $\boldsymbol{d}$ is the N vector of corresponding measurement. The minimal error can be found as:

$$e^2 = \boldsymbol{d}^T \boldsymbol{d} - \hat{\boldsymbol{\mu}}^T J^T J \hat{\boldsymbol{\mu}} \ . \tag{3.25}$$

It is also possible to include a term which models the knowledge that the solution is likely to be close to the predicted location, i.e. $\boldsymbol{\mu} \sim 0$:

$$\begin{aligned} \hat{\boldsymbol{\mu}} &= \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \ |J\boldsymbol{\mu} - \boldsymbol{d}|^2 + \gamma|\boldsymbol{\mu}|^2 && (3.26) \\ &= (\gamma I + J^T J)^{-1} J^T \boldsymbol{d} \ . && (3.27) \end{aligned}$$

Unfortunately, the measurements are not just subject to independent, Gaussian noise. Occlusion and misdetection mean that the errors are both correlated, and significantly non-Gaussian. In Drummond & Cipolla [1999], a robust M-estimator is used to model the non-Gaussian nature. M-estimators introduce non-Gaussian weights on the different measurements:

$$\hat{\boldsymbol{\mu}} = \underset{\boldsymbol{\mu}}{\operatorname{argmin}} \ \sum_{i=1}^{N} f(J_i \boldsymbol{\mu} - d_i) \ . \tag{3.28}$$

Zhang [1997] gives more information on M-estimators, but some for forms of $f()$, $\hat{\boldsymbol{\mu}}$ can be found using several iterations of *weighted* least squares. In Drummond & Cipolla [1999]:

$$\hat{\boldsymbol{\mu}} = (\gamma I + J^T W J)^{-1} J^T W \boldsymbol{d} \tag{3.29}$$

$$\text{where } W = \begin{bmatrix} w(d_1) & 0 & & 0 \\ 0 & w(d_2) & & 0 \\ & & \ddots & \\ 0 & 0 & & w(d_N) \end{bmatrix} \tag{3.30}$$

$$\text{and } w(d) = \frac{1}{c + |d|} \tag{3.31}$$

is used successfully.

Computationally, it is sensible to compute $\sum J_i^T w(d_i) J_i$ and $\sum J_i^T w(d_i) d_i$ instead of $J^T W J$ and $J^T W \boldsymbol{d}$ and to only calculate one half of the symmetric matrix $J^T W J$. Since $(\gamma I + J^T W J)$ is guaranteed to be positive definite, a Cholesky decomposition can be used to efficiently calculate its inverse. The Cholesky decomposition factorises a matrix into $A = LL^T$, where L is a lower triangular matrix. Then to solve $A\boldsymbol{x} = \boldsymbol{y}$ one first solves $L\boldsymbol{b} = \boldsymbol{y}$ by back substitution for $\boldsymbol{b}$ and then $L^T \boldsymbol{x} = \boldsymbol{b}$ by back substitution to find $\boldsymbol{x}$.

The Cholesky decomposition can be obtained by equating coefficients:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix} = \begin{bmatrix} l_{11} & 0 & \cdots & 0 \\ l_{21} & l_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{bmatrix} \begin{bmatrix} l_{11} & l_{21} & \cdots & l_{n1} \\ 0 & l_{22} & \cdots & l_{n2} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & l_{nn} \end{bmatrix} , \tag{3.32}$$

which then gives:

$$l_{ii} = \sqrt{\left(a_{ii} - \sum_{k=1}^{i-1} l_{ik}^2 \right)} \tag{3.33}$$

$$l_{ji} = \left(a_{ji} - \sum_{k=1}^{i-1} l_{jk} l_{ik} \right) / l_{ii} \ . \tag{3.34}$$

Since $A = \gamma I + J^T W J$, it is guaranteed to be symmetric and positive definite and hence the expression under the square root is positive, and the l_{ij} will be real.

3.2.5 Inlier/Outlier Ratio

The choice of optimisation is normally justified by a 'production model' which predicts how measurements are produced from the real world. A common approach [Torr & Zisserman 2000] assumes that real edges produce edgels which are disturbed from their true location by white noise, and additional edgels are produced randomly, with a uniform distribution over the image. A 'inlier/outlier ratio' is therefore needed to determine the proportions of the two types. There are two difficulties with this model; firstly the inlier/outlier ratio is not generally known *a priori* and hence must be estimated from the available data (see Section 2.4 and Tordoff & Murray [2002]). Secondly the optimisation function which results from this model is difficult to optimise.

The optimisation function described in Equation 3.31 gives a close approximation which does not suffer from these two difficulties. As before, the production model is that edges produce edgels disturbed by white noise and that edgels are also produced with a uniform distributed, except that now the second type cannot be placed within a certain width of an edge. The optimisation function still includes a 'penalty' term for edgels which are not explained by a model edge, but this is expressed in the width of the M-estimation function, and this can be physically estimated.

Chapter 4

Multi-Modal Tracking

4.1 Tracking Difficulties

The MAV environment described in Section 1.2 presents two characteristics which, when combined, represent a serious challenge to a visual tracking system. The problem starts with the poor quality of images obtained from the transmitting video camera. Blur, specularities, saturation, changing exposure and pixel noise all mean that the detection and matching of point-like features is problematic (see Section 4.4). Referring to typical images (e.g. Figure 1.2), the distinctive features are the large regions of consistent texture and the edge boundaries between them. It is evident that the presence of such edges is largely independent of viewpoint, lighting, and noise, and this property makes edge tracking the ideal technology for such images.

Unfortunately, while the presence of edges is very stable, their apperance is not. Not only does their appearance change with viewpoint and lighting variations, but more importantly, the appearance can vary radically *along* the edge. To match edges based on their appearance would therefore require a slow cross correlation evaluation along the length of each candidate match. Instead, previous edge-based systems have relied on accurate pose predictions and assume that the closest

detected edge to each predicted model line is the correct match. The second important characteristic of the MAV is that it is subject to relatively large motion disturbances and accelerations. These discontinuities, typically due to gusts, the ground effect (downdraft) or collisions with the floor, are very unpredictable and result in significant errors in pose predictions. The result is that the closest match assumption is often violated.

Occasional incorrect edge matches can often be tolerated through the use of a robust M-estimator (Section 2.4). Such techniques work very well when the outlier matches are drawn independently from a random distribution. This is typically the case when image point features are matched and very high outlier ratios can be tolerated [Rosten & Drummond 2005]. Unfortunately, when tracking edgel features, outlier errors tend to be highly correlated; if one edgel search finds an incorrect edge closer than the correct edge, there is a strong chance that its neighbours will as well. This leads to the optimisation finding a local minimum and tracking fails as shown in Figure 4.1.

Figure 4.1: Two examples of a single hypothesis edge tracking system 'locking' on to an incorrect edge and falling into a local minimum.

The problem of correlated edgel errors was addressed by Armstrong & Zisserman [1995] who first used RANSAC to robustly estimate the location of each model edge in the image (see Section 2.2.2 for more details). Here, this is extended by admitting multiple hypotheses for each line. This gives the resilience to local

minima demonstrated by particle filtering approaches whilst still providing real-time operation.

4.2 Multi-Modal Posterior Representations

The Random Sample Consensus paradigm, introduced by Fischler & Bolles [1981], has been widely used throughout the computer vision field to obtain robust estimates of model parameters. The prototypical example of its use is to obtain the parameters of a straight line which best fits a set of noisy measurements, M, containing outliers. The parameters describing the straight line through two measurement points selected at random are obtained. A consensus count for this hypothesis is then found by comparing, to a threshold value, the distance from the line to all the measurement points. The process is repeated N times and the hypothesis with the highest consensus is finally selected.

The consensus system proposed by Fischler & Bolles [1981] can be expressed as:

$$C(\boldsymbol{\theta}) = \sum_{m \in M} f(\,\mathrm{dist}(m, \boldsymbol{\theta})\,) \ , \tag{4.1}$$

where $\mathrm{dist}(m, \boldsymbol{\theta})$ is a function giving the distance from a measurement point m to the straight line defined by the vector $\boldsymbol{\theta}$ and:

$$f(d) = \begin{cases} 0 & (d < t) \\ 1 & (d > t) \end{cases} \ , \tag{4.2}$$

where t is the consensus range or threshold. Defined in this way, the hypothesis with the smallest consensus score is chosen. Torr & Zisserman [2000] observe that the performance of the system is improved if, at no extra computational cost, this last function is replaced with:

$$\tilde{f}(d) = \begin{cases} d & (d < t) \\ t & (d > t) \end{cases} \ . \tag{4.3}$$

When the consensus score is expressed in this way, RANSAC is being used to find the minimum of a function defined continuously over model parameters. Torr

& Zisserman [2000] show how (under certain assumptions) $\tilde{f}(d)$ can be used to obtain a consensus score which is the negative log likelihood that the hypothesis is correct. From this the posterior probability $P(\boldsymbol{\theta})$ can be easily calculated, as used in Section 4.2.1.

The approach proposed here borrows an idea from CONDENSATION and retains *all* of the hypotheses tested. Each is then treated as a particle in the representation of the posterior distribution. Suppose that a set of measurements is obtained which contains noisy points from more than one straight line, along with some outliers. RANSAC applied to this data set would only find the parameters of the line with the most corresponding measurement points (given a sufficiently large N). Instead, if the consensus scores from all N RANSAC hypotheses are retained, these can be used to give an approximation to the multi-modal posterior. This representation is beneficial if the next task is to draw samples from the posterior since this can be approximated by drawing from the stored hypotheses. A good approximation can be obtained with a relatively low N since there are typically many proposal points at or near the modes of P. However, this means that it is *not* correct to simply select hypothesis n with probability $p_n = P(\boldsymbol{\theta}_n)$.

If a posterior distribution to be represented is $P(\boldsymbol{\theta})$, and given a set of points $\Theta = \boldsymbol{\theta}_1 \ldots \boldsymbol{\theta}_N$ drawn from a (different) proposal distribution $Q(\boldsymbol{\theta})$, then draws from the posterior distribution can be approximated by resampling from Θ, where the probability of selecting sample n is:

$$p_n \propto \frac{P(\boldsymbol{\theta}_n)}{Q(\boldsymbol{\theta}_n)} \ . \tag{4.4}$$

Provided that $Q(\boldsymbol{\theta}) > 0 \quad$ for all $\boldsymbol{\theta}$ where $\quad P(\boldsymbol{\theta}) > 0$ this process converges to sampling from P as $N \to \infty$. In the case of multi-modal RANSAC, the proposals are those generated by random sampling of data tuples created by a process driven by P. The probability of obtaining the sample point, $Q(\boldsymbol{\theta}_n)$, is a function of the size of the consensus set (the number of data points within the consensus range of $\boldsymbol{\theta}_n$). If the size of the consensus set is c_n and the number of data samples needed to form a hypothesis is j, then Q is proportional to:

$$Q(\boldsymbol{\theta}_n) \propto \ {}^{c_n}\mathbf{C}_j \propto \frac{c_n!}{j!(c_n - j)!} \ , \tag{4.5}$$

provided that the same consensus set is obtained from any j-tuple subset. Although this is not strictly the case, in practice the approximation works well. Hence when sampling from a multi-modal representation of a posterior generated by RANSAC, each hypothesised sample, $\boldsymbol{\theta}_n$ should be chosen with a probability, p_n, given by:

$$p_n = K_1 \frac{P(\boldsymbol{\theta}_n)}{c_n \mathbf{C}_j} \text{ , with } K_1 \text{ chosen such that } \sum p_n = 1. \tag{4.6}$$

4.2.1 Application to Finding Straight Edges in an Image

Given an initial estimate for the location of an edge in an image, the task is to generate the posterior distribution of the true position of that edge, given the image data. Since the true edge is expected to be close to the initial prediction, it is convenient to express this posterior as a motion from this predicted location. Motions tangential to the edge direction are very difficult to detect accurately (due to the aperture effect) and so only motions normal to the edge direction are modelled. The posterior distribution is therefore defined over two model parameters r_1 and r_2, which describe the normal motion of the two endpoints of the line (see Figure 4.2). First, edgels along 1D perpendicular searches are obtained, either at points of maximal intensity gradient, or using the scheme described in Section 4.3. The ideas proposed in the previous section are then applied to find straight lines through these edgels.

The required posterior distribution describes the probability density that the corresponding straight edge in the image is at $\boldsymbol{\theta} = [r_1 \ r_2]^T$. An image edge is modelled as causing edgels which are displaced from a straight line by additive Gaussian noise. If $d_{n,s}$ is the distance along each perpendicular scanline s, from the intersection with the hypothesis edge $\boldsymbol{\theta}_n$ to the nearest edgel, a close approximation (see Section 3.2.5) to the negative log likelihood is:

$$C(\boldsymbol{\theta}_n) = K_2 + \sum_s \tilde{f}({d_{n,s}}^2) \ , \tag{4.7}$$

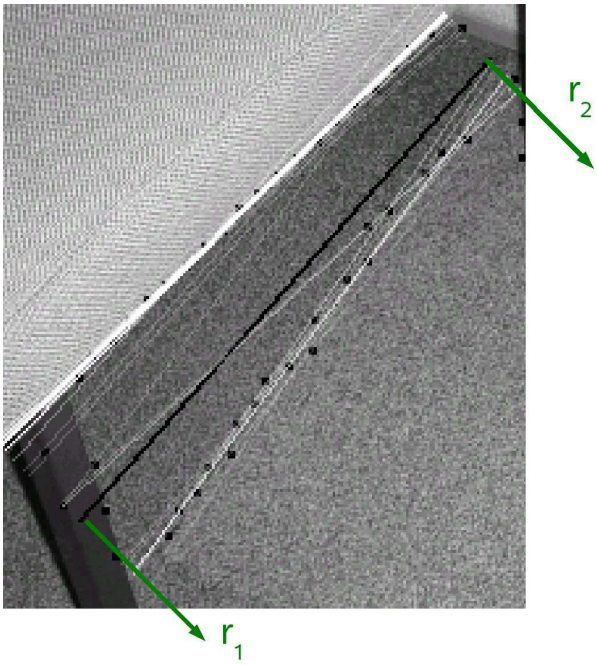


Figure 4.2: A closeup showing a rendered edge (black), the two model parameters, the detected edgels, and samples from the multi-modal representation (white).

with $\tilde{f}()$ as in Equation 4.3. Hence the posterior is given by:

$$P(\boldsymbol{\theta}_n) = K_3 \; e^{-\frac{C(\theta_n)}{2\sigma^2}} \quad . \tag{4.8}$$

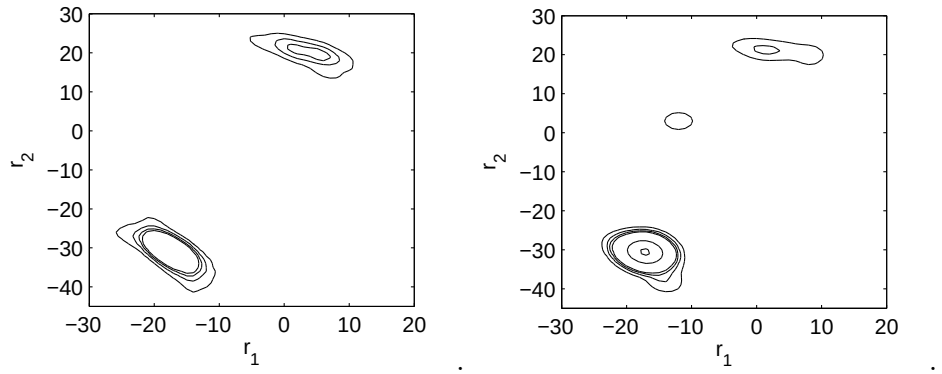
The N samples are obtained by randomly selecting two edgels ($j = 2$) and calculating the parameters that describe the straight line through these two points. The multi-modal representation is completed by calculating and storing $\Theta_n = \{\boldsymbol{\theta}_n, P(\boldsymbol{\theta}_n)\}$ for each. Since in this case $j = 2$, samples can then be drawn from the representation by selecting $\boldsymbol{\theta}_n$ with a probability proportional to:

$$p_n \propto \frac{P(\boldsymbol{\theta}_n)}{c_n(c_n - 1)} \quad , \tag{4.9}$$

where c_n is the corresponding consensus count as defined for Equation 4.5.

The behaviour of this approach is demonstrated in Figure 4.3 which compares contour lines of a full posterior distribution to that generated by sampling the RANSAC representation, with $N = 100$. To generate contour lines, the full distribution was calculated by dense evaluation, while the discrete 'impulse' samples

of the RANSAC respresentation were smoothed by convolution with a Gaussian. The RANSAC representation has both a similar shape to and scale as the full distribution, yet it is efficient to generate and samples from it can be easily made.



(a) Contours of the true posterior (b) The sampled representation

Figure 4.3: The multi-modal posterior for the edge shown in Figure 4.2.

4.2.2 Determining the Camera Pose

Having found a set of hypothesised correspondences for each model edge, a second level of RANSAC is used to determine the camera pose which maximises consensus of the original edgels. Three model edges are selected at random and for each a corresponding image line is chosen according to Equation 4.9. This is usually sufficient to constrain a new hypothesised pose. It is straightforward (see Section 3.2.3) to obtain a Jacobian matrix which describes the rate of change of each line parameter, r_i, with respect to each pose parameter, μ_j, about the current pose $\boldsymbol{\mu}$. Hence a linear approximation to the new hypothesised pose is given by:

$$\hat{\boldsymbol{\mu}} = \boldsymbol{\mu} + J^{-1}\boldsymbol{r} \quad \text{where} \quad J_{i,j} = \frac{\partial r_i}{\partial \mu_j} \;\; (i = 1..6, \;\; j = 1..6) \;\; . \tag{4.10}$$

The consensus score for the second stage RANSAC is found using a full Jacobian matrix to give a linear prediction for the new parameters of all the model lines. The probability density of each new line is evaluated from the original edgels using Equation 4.8 and the product of these then gives the overall pose posterior.

The second stage RANSAC is performed L times and the pose with the highest posterior is selected. This pose is finally optimised across all the consensus lines together (using each line's consensus edgels) to obtain the camera pose for that frame.

4.2.3 Relationship with Particle Filtering

The multi-modal representation proposed here is strongly related to that employed in particle filtering systems such as Isard & Blake [1998]. In both cases the true continuous posterior distribution is represented by measurements at a set of discrete sample points. In CONDENSATION, the assumption is that it is computationally cheap to measure the posterior at these sample points. Hence a relatively dense set of sample points can be generated at locations of high prior probability.

Unfortunately with a relatively complicated edge model, it is only possible to measure the posterior around 500 times per frame. Even if the prior for the frame has only one single peak, positioning 500 particles around this peak in six degrees of freedom gives just two or three measurements in each direction. This severly limits the motion disturbances which can be tolerated by traditional particle filtering schemes.

The system proposed here instead uses sample points which are directly suggested by the image data. Hence the particles are concentrated into areas which are likely to be near the peaks of the posterior. This significantly reduces the number of particles needed and allows larger motion disturbances to be tracked successfully.

4.3 Finding Texture Change-Points in a 1D Line Search

Traditionally [e.g. Harris & Stennett 1990; Isard & Blake 1998] an edgel is defined as a point of maximal intensity gradient along a 1D scanline. Here however edgels are considered as a change-point in a 1D texture process. Recent work [Shahrokni *et al.* 2004] describes a detector which finds the single most probable location of a texture change in a 1D scanline. Here this is extended to allow multiple possible change-points. This allows the tracking system to operate in simply textured environments (where otherwise tracking would fail) yet adds little extra computational cost.

Shahrokni *et al.* [2004] propose a tracking system similar to Harris & Stennett [1990] but use a 1D texture change-point search instead of a simple gradient based edgel detector. Pixel intensities are first grouped into regularly spaced intensity bins. Consider a known zeroth-order texture generating process T_1 where pixel intensities are independently drawn from a probability distribution over I intensity bins ($T_1 = \{p_i\}; i = 1..I$). Hence given a sequence of N binned pixel intensities $S_0^N = (s_0, s_1, ...s_{N-1})$, the probability of obtaining that sequence, given process T_1 is:

$$P(S_0^N|T_1) = \prod_{n=0}^{N-1} p_{s_n} \ , \tag{4.11}$$

The key result from Shahrokni *et al.* [2004] is that if the texture process is instead unknown, but all possible texture processes are equally likely (i.e. there is a uniform prior on T), then given a sequence of n pixel intensities $S_0^n = (s_0, s_1, ...s_{n-1})$ from T, the expected probability of a further sample, s_n from T is given by:

$$E(p_{s_n}|S_0^n) = \frac{o_{s_n} + 1}{n + I} \ , \tag{4.12}$$

where there are o_j occurrences of the symbol j in the sequence S_0^n. Hence the probability of the entire sequence given a single texture is:

$$P(S_0^N) = \prod_{i=0}^{N-1} E(p_{s_i}|S_0^i) \quad = \frac{(I-1)! \ \prod_{i=1}^{I}(o_i+1)!}{(I+N-1)!} \ , \tag{4.13}$$

In Shahrokni *et al.* [2004] this result is used to find the most probable location for a single texture change-point along a scanline and an efficient algorithm for this was presented. In this work however, multiple change-points must be considered. Instead of maximising the probability of the location of the single change-point given a sequence of intensities, the set of change-points with maximum probability must be found. If the sequence of N pixels, S, is modelled by $m-1$ distinct textures between m change-points, $M = (c_1, c_2 ... c_m)$:

$$P(S|M) = \prod_{j=1}^{m} P(S_{c_j}^{c_{j+1}}) \ . \tag{4.14}$$

Finally using Bayes' rule:

$$P(M|S) = \frac{P(S|M)P(M)}{K} \tag{4.15}$$

is maximised. For this work $P(M)$ was chosen simply as $P(M) = \lambda^m$, with λ constant and < 1.

Evaluating Equation 4.15 for all 2^N possible combinations of change-points would be computationally prohibitive. Fortunately it is possible to use a dynamic programming algorithm to improve efficiency greatly. The most probable set of change-points can be determined by considering the location of the last change-point in the sequence, c_m. Suppose that $\text{argmax}_{M'} P(S_1^k|M')$ is known for all $k < l$. The most probable position of c_m is then:

$$\begin{aligned} c_m &= \underset{c}{\text{argmax}} \ P(S_0^l|\text{last change-point at } c) \\ &= \underset{c}{\text{argmax}} \ \left[\max_{M'} P(S_0^c|M')P(S_c^l|\emptyset)\right] \ , \end{aligned} \tag{4.16}$$

since the probability of the sequence after c_m is independent of the sequence up to c_m. $\emptyset$ is the empty set implying no change-points. Hence:

$$\begin{aligned} \underset{M}{\text{argmax}} \ P(S_0^l|M) &= \{l\} \cup \underset{M'}{\text{argmax}} \, P(S_0^{c_m}|M') \\ \text{and} \qquad \max_{M} \ P(S_0^l|M) &= \max_{M'} P(S_0^{c_m}|M')P(S_{c_m}^l|\emptyset) \ . \end{aligned} \tag{4.17}$$

By induction these expressions can be evaluated sequentially for $l = 0..N$.

This means that, at the first step where $l = 0$, $P(S_0^0|0,1)$ is evaluated using s_0. At the second step ($l = 1$) both $P(S_0^1|0,2)$ and $P(S_0^1|0,1,2)$ must be evaluated using the value from the first step and s_1. Using these values, there are four potential sets of change-points to evaluate at the third step. However, the change from $P(S_0^1|0,2)$ to $P(S_0^2|0,2,3)$ is the same as that from $P(S_0^1|0,1,2)$ to $P(S_0^2|0,1,2,3)$ since by Equation 4.14, the probability of the sequence after a change-point at 2 is independent of the sequence before it. Hence the most probable of $P(S_0^1|0,2)$ and $P(S_0^1|0,1,2)$ will lead to the most probable of $P(S_0^2|0,2,3)$ and $P(S_0^2|0,1,2,3)$. So the third step simply involves picking the highest probability from the second step, and it is only necessary to evaluate three new probabilities. This means that when processing the n^{th} pixel only n possibilities must be evaluated, giving a total of $N^2/2$ operations for the entire sequence. The complete operation is shown in Algorithm 1.

Whilst this is not a major computational load, a very simple speed-up is possible. Since (see Appendix A):

Theorem 1 *Equation 4.15 will never be maximal if there is a change-point between pixels of the same binned intensity*

there is no need to consider such cases. This offers a speed improvement in areas of the image which have a relatively constant intensity.

4.4 Results

The tracking system was tested on two full video sequences, one of a staircase and one of a corridor. Although both sequences were prerecorded, processing was performed at 10fps and 30fps respectively. Although a more complicated motion model could have been used, for testing purposes, in both cases the predicted location for each frame was simply that found in the previous frame. This increases the challenge to the tracking system since it must correct for any motion, not just any acceleration.

Algorithm 1 Finding texture change-points along a 1D scanline

```
dim Observations[NUM_PIXELS][NUM_BINS]
dim LogLikelihoods[NUM_PIXELS], Changepoints[NUM_PIXELS]
BestLogLikelihood=0

{Loop for each pixel in the sequence}
for i=1..NUM_PIXELS do

  {Set up the new node with a cleared histogram}
  for j=1..NUM_BINS do
    Observations[i][j]=1     {Seed with 1 sample per bin}
  end for

  {Start with the best log likelihood to that change-point plus prior penalty}
  LogLikelihoods[i]=BestLogLikelihood-log(λ)

  {Loop round the nodes currently in the tree}
  BestLogLikelihood=BIG_NUM
  for k=1..i do

    {Update the node's log likelihood given the new pixel using Equation 4.12}
    LogLikelihoods[k]+=log( ∑Observations[k][] )
                          - log( Observations[k][Pixels[i]] )
    Observations[k][Pixels[i]]+=1

    {and store the best ready for the next outer loop}
    if LogLikelihoods[k] < BestLogLikelihood then
      BestLogLikelihood=LogLikelihoods[k]
      Changepoints[i]=k
    end if

  end for
end for

{The optimal change-points can now be found by working back:}
{c1=Changepoints[NUM_PIXELS], c2=Changepoints[c1], etc.}
```

4.4.1 Staircase Sequence

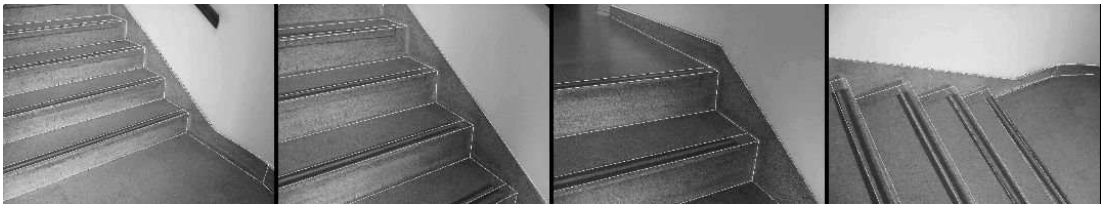
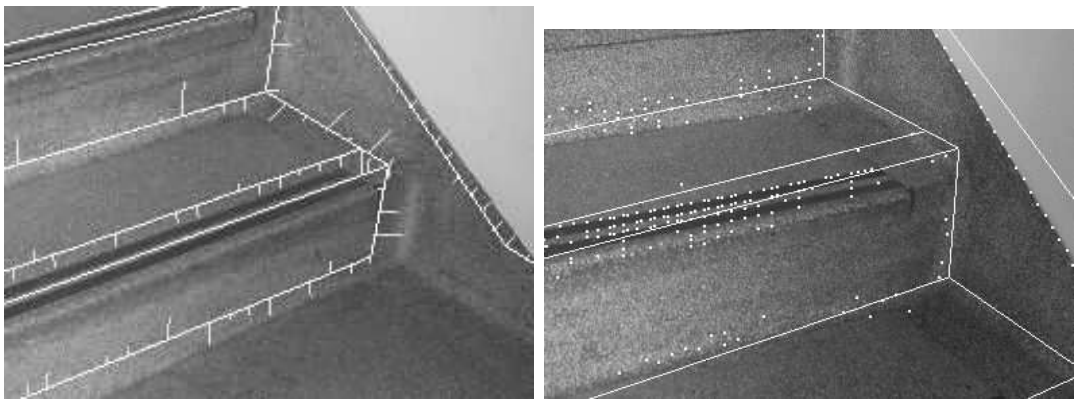


Figure 4.4: Frames 1,50,100,150 from a staircase video.

The staircase sequence (Figure 4.4) demonstrates the system's ability to cope with textured surfaces as well as rapid motion (which leads to large inter-frame changes since no motion model is used) and significant orientation changes. This sequence was recorded with a digital camera since the MAV transmitting camera was not yet available. Attempts to track the sequence using an earlier edge based system [Drummond & Cipolla 1999] fail almost instantly as shown in Figure 4.5. Tracking also fails if a simple gradient based edge detector is used instead of the scheme proposed in Section 4.3. Since the speckled steps produce a huge number of hypothesis edgels, a very large number of RANSAC hypotheses (for both stages) would be needed to track successfully. Using the texture detector proposed in Section 4.3 makes the task feasible and the sequence tracks correctly.



(a) A closeup of frame 10 from the staircase sequence showing the earlier system failing to find the correct edges.

(b) Using multi-hypothesis RANSAC also fails if a simple gradient based edge detector is used.

Figure 4.5: Failures of previous systems on the staircase sequence.

4.4.2 Corridor Sequence

The corridor sequence uses relatively poor video, such as might be expected from a miniature transmitting camera. Some frames showing the successful tracking of the sequence are shown in Figure 4.6. Tolerance of significant pixel noise and blurring, and robustness to occlusion, is demonstrated.

The corridor sequence was also used to test two other tracking systems: a demonstration version of 'boujou' and the earlier edge-based system [Drummond & Cipolla 1999][1]. 'boujou' is a commercial product (`www.2d3.com`) which performs offline bundle adjustment of tracked interest points. For a simple quantitative comparison, the number of tracking failures suffered by each method was recorded for three different sequences from the same corridor. After each failure the pose was manually reset.

Table 4.1: Results from tests on three corridor sequences, showing the number of tracking failures for the three different schemes.

		Number of Failures		
Seq.	No. frames	boujou	Earlier Edge Based	Multi-modal RANSAC
1	623	18	13	0
2	763	35	21	6
3	692	22	8	3

The blur and pixel noise present in the images means that reliable point features are difficult to obtain and match, as shown in Figure 4.7. Hence boujou was generally unable to maintain track for more than a few frames at a time.

[1] To take advantage of more powerful computing now available, ten complete tracking iterations were performed per frame. This enables the M-estimator to converge fully at each frame.

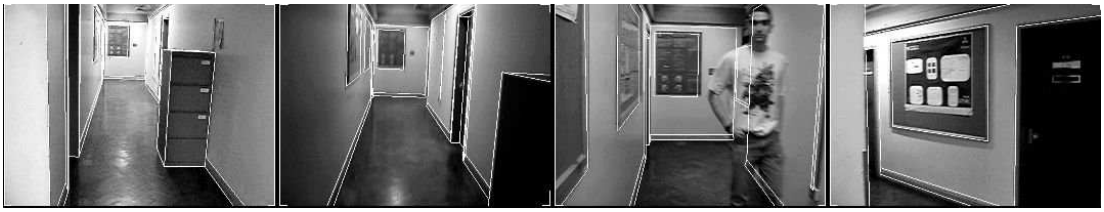
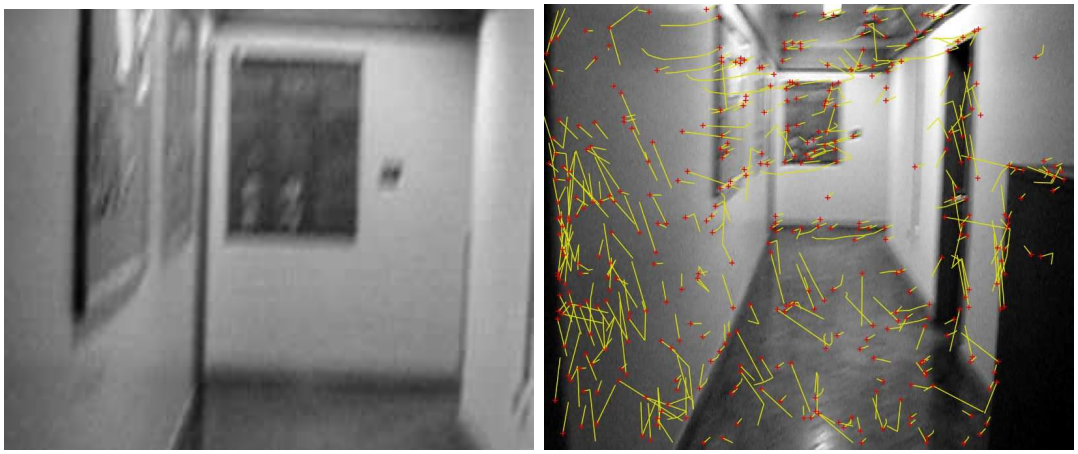


Figure 4.6: Frames 1,200,400,600 from a corridor sequence, being correctly tracked by the multi-modal RANSAC system.



(a) A closeup showing typical blur levels.

(b) Reliable point matching is difficult with this level of blur and noise.

Figure 4.7: Difficulties tackled in the corridor sequence.

4.5 Conclusions

This chapter presented a method for generating a representation of a multi-modal distribution. This is particularly beneficial when partitioning a problem such that the multiple hypotheses from one stage provide samples for further processing. The method has been successfully applied to the task of locating straight edges in an image using the results from a novel texture change-point detector and subsequently to perform visual tracking of known polyhedral objects. Due to the data-driven approach, the system shows a significant improvement in motion robustness over previous real-time techniques.

There remain two areas for improvement. The texture change-point detector provides a useful improvement over a simple gradient based detector. However,

due to the binning of pixel intensities, the algorithm fails to account for the intensity change near edges. In reality, a scanline of intensities which smoothly ramps across a bin boundary is far less likely to be an edge than one which jumps up several bins, but by Equation 4.11 these two cases are equally likely. The problem was avoided for this implementation by not allowing change-points to occur when the intensity only changes across one bin boundary. In this way, the detection of unlikely edgels is avoided and indeed, with appropriate parameter choices, it is possible to degrade the system to become a simple gradient detector. However, the problem indicates an inaccuracy in the underlying probability model and this should be addressed more rigorously.

The hierarchical RANSAC approach works well in the majority of cases. However, by nature, RANSAC relies on being able to find the correct consensus set from just three edge-to-line matches. In practice, the majority of tracking failures observed occured not because RANSAC failed to try good combinations but because (even if selected manually) *no* three matches existed to adequately constrained the pose. Of course, this problem could be mitigated if more than three matches were tested by RANSAC but this would result in significantly sub real-time operation. The next chapter describes a better alternative.

Chapter 5

Dynamic Measurement Clustering

The work described in the previous chapter provided a significant improvement in tracking reliability over previous approaches. Much benefit was obtained through the use of a divide-and-conquer technique. Such hierarchical approaches determine, in a first pass, a subset of the parameters. Subsequent passes then expand this subset until the entire set of parameters is computed. The advantage of these techniques is that each pass is presented with a search problem of reduced dimensionality; for problems with noisy data this can result in substantial computational savings. The divide-and-conquer approach has been applied to both particle filtering [MacCormick & Isard 2000] and continuous optimisation problems [Marchand *et al.* 1999; Rasmussen 2004] with great success. However, in all these cases, the partitioning is statically predetermined by the programmer.

This chapter presents a method for automatically partitioning the search space. This yields the benefit that the system can be applied to problems, such as rigid body pose estimation, that do not admit a constant partitioning of the parameter space. In this case the system provides a *dynamic* partitioning which is locally optimised for the current conditions at each time step.

5.0.1 Overview of the Approach

The approach makes use of automatic partitioning by grouping feature measurements into clusters. Although the space of all possible pose changes may have a high dimensionality (≥ 6), the image motion of each feature will be affected by just a subspace of this parameter space[1]. More precisely, there is a parameter subspace - its null space - that does *not* affect the position of the feature in the image. By grouping, at each frame, trackable features which share such a subspace, the pose estimation problem can be decomposed into a number of lower-dimension problems that can be solved in parallel.

One of the greatest difficulties when estimating the pose of the MAV is obtaining the correct set of correspondences between model features and image features. Fortunately, the clustering process is independent of any correspondence assumptions. Within each cluster, there is therefore the opportunity to evaluate correspondence hypotheses in a reduced dimension coordinate frame and this provides a significant computational reduction.

The next section describes the method by which clusters can be obtained, both for the general case and for the specific application of edge tracking. Section 5.2 shows hows how the solution(s) for each cluster sub-problem can be found and then combined to give a robust solution to the whole pose estimation problem. For the MAV tracking problem, the computational complexity is reduced to the point where an exhaustive search can be used within each cluster in real-time. It is expected that similar benefits would apply if other algorithms were used to solve for each cluster.

[1]This subspace will not necessarily be axis-aligned.

5.1 Clustering

5.1.1 Corner-Based Example

Consider the task of estimating the six dof camera pose change between the two images shown in Figure 5.1, using feature point correspondences. Assume that the 3D positions of the corners in the top image are known. A natural approach to solving this problem as a human viewer is to first match the building in the background and then independently to locate the fountain in the foreground. Separating the problem into two in this way is successful because the building is sufficiently far away that its change in appearance is essentially independent of camera translation and depends only on camera rotation. Thus, using just matches for points on the building, the 3-dimensional camera rotation problem can be solved in isolation. Similarly for the fountain (which is closer), lateral and vertical motions of the camera look rather like rotations of the camera about vertical and lateral axes respectively. Thus its image motion can be well approximated by just four parameters.

The example shown in Figure 5.1 demonstrates several key features of the clustering approach.

1. Computational benefits when solving the correspondence problem:
   Because the building has much repeated structure, it is difficult to find the correct match for each feature point using local image information. For the small selection of points features shown in Figure 5.1 (top), the best three matches in the second image (by sum squared image patch difference) were needed in order to include the correct match. By grouping features into separate clusters for the building and for the fountain, the computational problem of selecting the correct matches is reduced. For example, using RANSAC, only two points need to be randomly chosen to determine the motion of the fountain and separately only pairs of points (with a constraint) are chosen to locate the building. Without clustering, sets of at least three

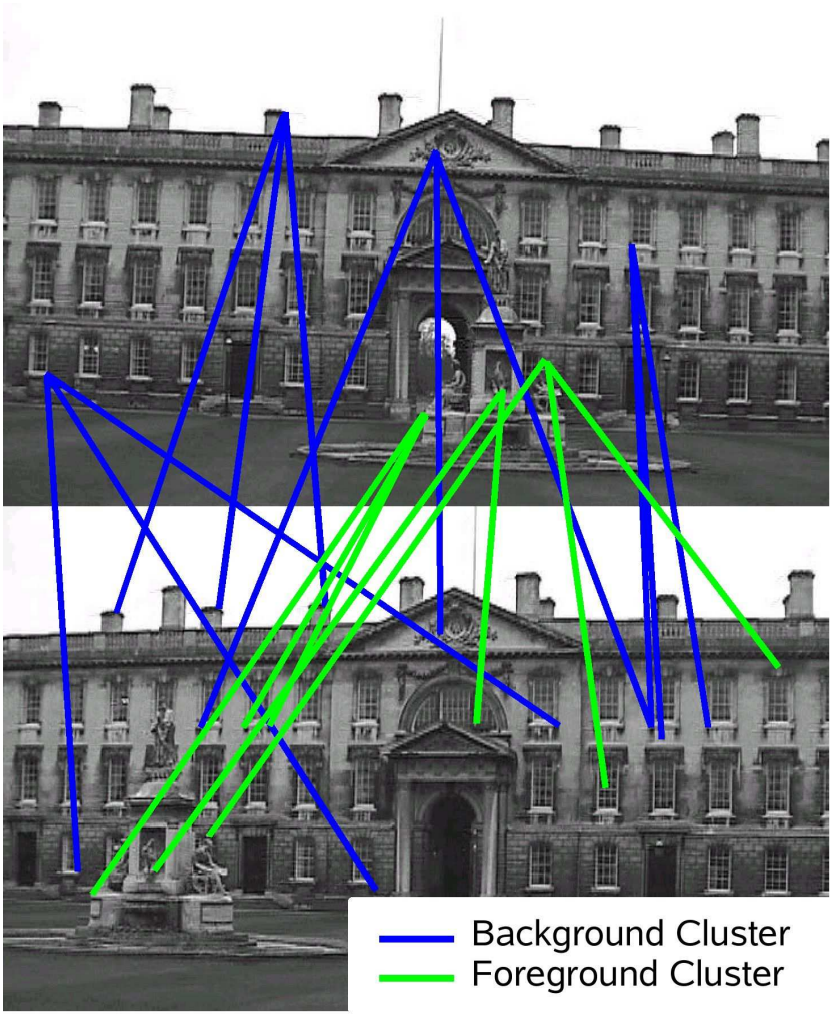


Figure 5.1: When trying to track the change in pose between the top and bottom frame it is sensible to consider a cluster of all corner matches for the fountain, separately from a cluster containing corners on the building.

points would have to be randomly chosen and this is significantly more computationally expensive.

2. Hypothesis verification:
   For feature clusters to be useful it must be possible to verify hypotheses within the cluster. For example, two points need to be chosen to determine the motion of the fountain. If however the fountain cluster contained only two features, there would be no means to score or order the hypotheses and the results are as ambiguous as the original situation. Hence it is vital that a cluster contains more measurements than are needed to simply constrain the hypotheses so that the cluster can be used to propose a small set of likely possibilies.

3. Weak constraint:
   Although the change in appearance of the building is essentially independent of camera translation (i.e. camera translations are in the null space of the cluster), this would strictly only be true if the building were infinitely far away. In practice, null space parameter changes will actually affect each measurement with a small error. However, a cluster is only used to suggest a small set of likely hypotheses for the correspondence sets. Hence it is only important that incorrect correspondences are not introduced by any such errors. This is enforced using a prior assumption that the parameter error (unpredicted pose motion) is limited to size k_m and that correspondence errors will not be caused by measurement errors less than k_c. These parameters are discussed further in Section 5.1.7.

4. Maximum error:
   The following sections include a description of how to find cluster centres which minimise a sum-squared error metric. However, as just described, it is important that no correspondence errors are introduced and hence the relevant quantity for determining the validity of a cluster is not the sum-squared error but rather the maximum error of any of the measurements in the cluster.

5. Verification of combinations:
   Having obtained a small number of hypotheses for each cluster, to obtain the overall camera pose it is necessary to evaluate combinations of these hypotheses. Many of these combinations can be quickly rejected since a combination will only be valid if the individual hypotheses agree on any shared parameters. In the building and fountain example, a total of seven parameters are obtained from combining individual results, yet there are only six parameters to be determined. The common parameter is (roughly) camera rotation about its optical axis; unless the fountain has fallen over, both clusters should agree on this parameter.

5.1.2 Mathematical Preliminaries

Consider the problem of estimating a parameter vector $\boldsymbol{\mu}$ in an N dimensional space ($\boldsymbol{\mu} = [\mu_1 ... \mu_N]^T$). Suppose there are M one-dimensional model features. Without yet imposing correspondences between model and image features, a $M \times N$ Jacobian matrix J can be formed to describe the partial derivatives of each measurement error with respect to each of the parameters:

$$J_{mn} = \frac{\partial d_m}{\partial \mu_n} \quad . \tag{5.1}$$

If the measurements are only subject to independent Gaussian noise, a least squares solution would give the solution to:

$$\underset{\boldsymbol{\mu}}{\operatorname{argmin}} \; |J\boldsymbol{\mu} - \boldsymbol{d}|^2 \quad . \tag{5.2}$$

If however correspondence errors lead to $\boldsymbol{d}$ containing many erroneous measurements, a robust technique is typically used. RANSAC can be used to find:

$$\underset{\boldsymbol{\mu}\,,\,\mathsf{V}}{\operatorname{argmin}} \; |J_{\mathsf{V}}\boldsymbol{\mu} - \boldsymbol{d}_{\mathsf{V}}|^2 - k_p\,|\mathsf{V}| \quad , \tag{5.3}$$

where V is the set of valid measurements, $|\mathsf{V}|$ is the cardinality of the set, k_p is a penalty per erroneous measurement and subscripting with a set (J_{V} and d_{V}) selects only the corresponding rows from the original matrix or column vector.

Unfortunately, in high-dimensional parameter spaces, RANSAC becomes computationally expensive; in the exhaustive case it has order M^N. Here RANSAC or exhaustive searching is instead used within clusters of measurements and this significantly reduces the overall computational cost.

5.1.3 Measurement Clustering

Clustering aims to group together measurements which depend only on a subspace of the parameter space; hence a cluster is a set C of rows of J which have:

$$\operatorname{rank}(J_{\mathsf{C}}) < N \ . \tag{5.4}$$

Suppose the set of M measurements can be exactly split into C_1 and C_2, so that Equation 5.3 can be rewritten as:

$$\underset{\boldsymbol{\mu},\mathsf{V}_1\in\mathsf{C}_1,\mathsf{V}_2\in\mathsf{C}_2}{\operatorname{argmin}} \begin{pmatrix} |J_{\mathsf{V}_1}\boldsymbol{\mu} - \boldsymbol{d}_{\mathsf{V}_1}|^2 - k_p\,|\mathsf{V}_1| & \text{(A)} \\ +\,|J_{\mathsf{V}_2}\boldsymbol{\mu} - \boldsymbol{d}_{\mathsf{V}_2}|^2 - k_p\,|\mathsf{V}_2| & \text{(B)} \end{pmatrix} \tag{5.5}$$

Now suppose that the partitioning can be chosen such that the clusters are independent and that M is large so that $\operatorname{rank}(J) = N$, then:

$$\operatorname{rank}(J_{\mathsf{C}_1}) + \operatorname{rank}(J_{\mathsf{C}_2}) = N \ . \tag{5.6}$$

For this didactic case, (A) and (B) of Equation 5.5 provide independent constraints on $\boldsymbol{\mu}$ and hence V_1 and V_2 can be found separately. This gives a large computational saving; for the case of exhaustively searching all combinations, the computation order reduces from M^N to as little as $2(M/2)^{(N/2)}$.

In general of course, the measurements will not fit into independent clusters and instead:

$$N < \operatorname{rank}(J_{\mathsf{C}_1}) + \operatorname{rank}(J_{\mathsf{C}_2}) < 2N \ . \tag{5.7}$$

In this case, (A) and (B) of Equation 5.5 no longer provide independent constraints on $\boldsymbol{\mu}$. However, it can be seen that it is straightforward to solve Equation 5.3 for $\boldsymbol{\mu}$ once the optimal set of inlier measurements, V, has been found.

Clustering works by assuming that the optimal solution for V (and hence $\boldsymbol{\mu}$) can be obtained by combining the optimal solutions for $\mathsf{V}_{\mathsf{C}_1}$ and $\mathsf{V}_{\mathsf{C}_2}$ as found by considering (A) and (B) of Equation 5.5 as still independent.

The clustering assumption is aided by two factors. Firstly, as discussed in Section 5.1.1 (item 2), it is important that for each cluster:

$$\text{rank}(J_{\mathsf{C}}) < |\mathsf{C}| \ , \tag{5.8}$$

since this ensures that solutions obtained independently within a cluster can be meaningfully evaluated by testing consensus within the cluster. Secondly, if the best result from each cluster agrees with those from the other clusters on all overlapping dimensions of $\boldsymbol{\mu}$ (see Section 5.1.1, item 4) then the optimal solution to Equation 5.3 will be found as this combination. In other cases it is entirely feasible to consider the best *few* solutions for each cluster and to compare the possible combinations by applying Equation 5.3.

5.1.4 Weakening the Rank Constraint

Although there are some special cases where Equation 5.4 and Equation 5.8 can be simultaneously satisfied, in general this is not strictly possible. However, as discussed in Section 5.1.1 (item 3), it is reasonable to weaken the precise rank constraint, provided that the correct set of inliers V for each cluster can still be obtained.

Consider a cluster C which has *approximate* rank w. The object is to find the optimal point in the w-dimensional subspace of $\boldsymbol{\mu}$ (the cluster's search space, $\mathcal{S}$). Given such a point $\boldsymbol{\mu} \in \mathcal{S}$, inclusion of each measurement into the inlier set is determined by the consensus test dictated by Equation 5.3:

$$i \in \mathsf{V} \ \text{ if } \ |J_i\boldsymbol{\mu} - \boldsymbol{d}_i|^2 < k_p \ . \tag{5.9}$$

Now, if probable parameter changes outside $\mathcal{S}$ (those in the cluster's null space, $\mathcal{N}$) cause large changes in the value of $J_i\boldsymbol{\mu}$, then the results of this test will be

incorrect. However, if for all measurements in the cluster, only errors much less than k_p are caused by null space parameter changes, the inlier set proposed by the cluster will be valid.

In other words, a bounded motion in the null space must have a limited effect on each individual measurement in the cluster. For any parameter motion in $\mathcal{N}$ limited to size k_m, the maximum change, $m_i()$, in the ith measurement can be found by projecting J_i into $\mathcal{N}$:

$$\begin{aligned} m_i(\ \{\boldsymbol{\mathcal{S}}_1 ... \boldsymbol{\mathcal{S}}_w\}\ ) &= k_m \left| \sum_{j=1...N-w} \boldsymbol{\mathcal{N}}_j\ (\boldsymbol{\mathcal{N}}_j\ \cdot J_i) \right| \\ &= k_m \left| J_i - \sum_{j=1...w} \boldsymbol{\mathcal{S}}_j\ (\boldsymbol{\mathcal{S}}_j\ \cdot J_i) \right| \quad , \end{aligned} \tag{5.10}$$

where the vector $\boldsymbol{\mathcal{S}}_i$ is the ith orthonormal basis vector of the space $\mathcal{S}$. The second version is useful for the clustering algorithm described shortly and is valid since $\mathcal{S}$ is the space orthogonal to $\mathcal{N}$. Using this, a cluster is valid if for a threshold k_c:

$$m_i < k_c \ \ \forall\ i \in \mathsf{C}\ . \tag{5.11}$$

5.1.5 Finding Clusters

A simple greedy algorithm can now be used to form the clusters. Since clusters with smaller w give greater computational advantages, clusters are formed with increasing w. The algorithm proceeds by considering each measurement in turn and seeding a cluster from it. This means that $\mathcal{S}$ is defined by a single row of J (and $\mathcal{N}$ is everything orthogonal to it). All the other measurements are then tested to see if they satisfy Equation 5.11 (for this $\mathcal{N}$) and the size of the cluster is computed. Having considered a cluster seeded from each measurement, the cluster with the largest size is then extracted. This is repeated until the size of the largest cluster is smaller than a threshold. At this point clusters seeded from each remaining *pair* of measurements are considered, followed by triples if needed and so on. This method is further described in Section 5.1.6.

Given a set of measurements which form a valid cluster, it is useful to find the search and null spaces. The largest sum squared error over any unit size parameter change in $\mathcal{N}$ is given by:

$$e(\mathcal{N}) = \max_{\boldsymbol{\mu}\in\mathcal{N},|\boldsymbol{\mu}|=1} \boldsymbol{\mu}^T J_{\mathsf{C}}^T J_{\mathsf{C}} \boldsymbol{\mu} \ . \tag{5.12}$$

Using this, the optimal common null space is:

$$\operatorname*{argmin}_{\mathcal{N}} e(\mathcal{N}) \ , \tag{5.13}$$

which can be found by calculating the eigenvalue decomposition of $J_{\mathsf{C}}^T J_{\mathsf{C}} = UDU^T$, where D is a diagonal matrix of descending ordered eigenvalues. From Equation 5.12, it can be seen that the optimal null space is spanned by the last $N - w$ columns of U:

$$U = \begin{bmatrix} : & : & : & : & : & : \\ \boldsymbol{\mathcal{S}}_1 & ... & \boldsymbol{\mathcal{S}}_w & \boldsymbol{\mathcal{N}}_1 & ... & \boldsymbol{\mathcal{N}}_{N-w} \\ : & : & : & : & : & : \end{bmatrix} \ . \tag{5.14}$$

The space spanned by the first w columns of U is orthogonal to $\mathcal{N}$ and describes the search space for the cluster, i.e. the subspace of parameters for which the cluster will yield solutions. The $(w + 1)$th largest eigenvalue gives the value of $e(\mathcal{N})$ for that space.

5.1.6 Line Feature Clustering

Since the video feed from the MAV is often blurred, noisy and devoid of point features, the remainder of this chapter discusses edge features. In this section, the process of obtaining good clusters of edges is described. The clustering process should find groups of model lines which are affected by only a subspace of the possible camera motions.

A set of straight line segments $\boldsymbol{l}_{1..L}$ is obtained by rendering the model at the predicted pose, as described in Section 3.1.2. Each line has two endpoints but because of the aperture effect only their motions in the direction normal to the

line can be measured reliably. Hence, unlike the description given in Section 5.1, each measurement is two-dimensional. Possible motions of the camera about the predicted pose are given by the six parameters $\boldsymbol{\mu} = [\mu_1 \ \mu_2 \ ... \ \mu_6]^T$ and so by differentiating about the current pose, a $2L \times 6$ Jacobian matrix J can be found (see Section 3.2.3). Camera motions which are in the space spanned by a line's two Jacobian vectors will cause a (normal) motion of that line in the image. Conversely, camera motions contained within the four-space orthogonal to these two vectors, will not cause any motion of the line.

The task is to form valid clusters (as given by Equation 5.11) which contain as many line segments as possible. As briefly described above, a greedy two-pass algorithm is used. During the first pass, L test clusters centred on each line segment are created using Gram-Schmidt orthogonalisation of the relevant two Jacobian rows. For the c^{th} cluster:

$$\begin{aligned} \boldsymbol{\mathcal{S}}_1^c &= \text{normalise}(\ J_{2c}\ ) \\ \boldsymbol{\mathcal{S}}_2^c &= \text{normalise}(\ J_{2c+1} - \boldsymbol{\mathcal{S}}_1^c(J_{2c+1} \cdot \boldsymbol{\mathcal{S}}_1^c)\ ) \end{aligned} \tag{5.15}$$

The maximum size of each test cluster is found using Equation 5.10:

$$\text{size}(c) = \sum_{i=1..L} \begin{cases} 1 & \text{if} \quad m_i(\ \{\boldsymbol{\mathcal{S}}_1^c \ \boldsymbol{\mathcal{S}}_2^c\}\ ) < k_c \\ 0 & \text{otherwise} \end{cases} \tag{5.16}$$

and clusters are then sorted by size. The lines from the largest cluster are extracted and the optimal cluster centre found using Equation 5.14. The test cluster sizes are updated with these lines omitted and the process is repeated.

It is perfectly acceptable for two different clusters to contain the same line segment and in many cases allowing this will improve the reliabilty of the results from each cluster. However, computationally it is also important that clusters are reasonably unique; creating a new cluster which differs only in a few edges from another cluster is unlikely to be beneficial. Hence is is important to obtain a compromise between the number of different clusters and the number of edges in each cluster. Although more complicated measures or constraints could be employed, limiting each edge to be a member of just one cluster appears to produce satisfactory results for the cases tested.

A first pass of the algorithm finds 2-clusters of model edges – clusters with a two-parameter search space and four-parameter null space. As discussed in Section 5.1.1, for a 2-cluster to be useful it must contain at least two edges and preferably many more; the clustering process is therefore halted when new clusters would contain fewer than three edges. The remaining edges are then clustered into 4-clusters, which will typically accept many more edges, using a similar algorithm to that just described.

5.1.7 Clustering Results

This section demonstrates the results of applying the clustering algorithm to two different edge models. The first example is a model of a corridor used for flying the MAV. The second is a model of a maze-like structure used for an augmented reality application.

Choosing the ratio of maximum camera motion (k_m) to clustering error (k_c) is a compromise between finding good clusters and limiting the possibility for correspondence errors. For both examples a maximum camera motion (k_m) of 0.15m or 0.3rad is used. With an ideal constant velocity filter predicting the camera motion at each frame, this maximum would correspond to permitting translational accelerations of up to ~ 27g; substantially more than experienced by the MAV. Section 6.1.1 discusses the filtering difficulties which demand a much larger permitted camera motion than expected.

For k_c a value of 4 image pixels is chosen. Larger values would increase the number of edges in each cluster and hence reduce the number of clusters and the computational load. However, larger values mean that correspondence errors may occur if the camera motion in the cluster null space happens to be large. A value of 4 pixels has the effect that if there are two lines in a cluster which are less than 8 pixels apart, the correct hypothesis from the cluster may be missed or scored poorly. However, in practice, large unexpected motions which create close

to 4 pixel errors almost always result in significant image blur and it is unlikely that close lines will be separately detected.

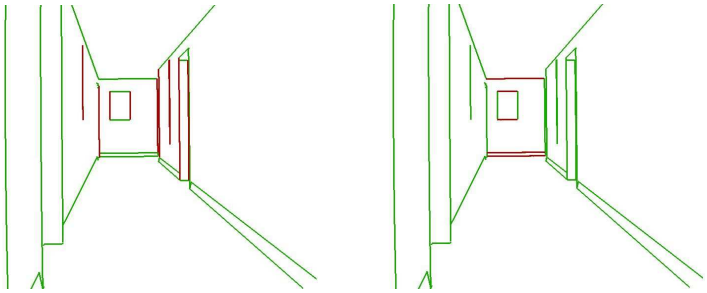
Figure 5.2 shows the clustering results for a particular pose of each of the two scenes. The corridor scene contains significant depth and the clustering makes good use of this by allocating 2-clusters to parallel edges in the distance. These edges are virtually unaffected by camera translation and so can very effectively be used to obtain the rotational pose parameters. The maze scene is much closer to the camera and 2-clusters are confined to edges which are physically close together. 4-clusters are very useful in this case.

On the 2.6GHz machine the entire clustering operation typically takes less than 1ms. However, due to the formation of 4-clusters, the algorithm presented has cubic complexity in the number of model lines and for some poses of the maze scene the computation time becomes significant. In these cases it is possible to iteratively re-cluster based on the cluster centres from previous frames. Only occasional $O(L^3)$ operations are then necessary to find new cluster centres as they appear.

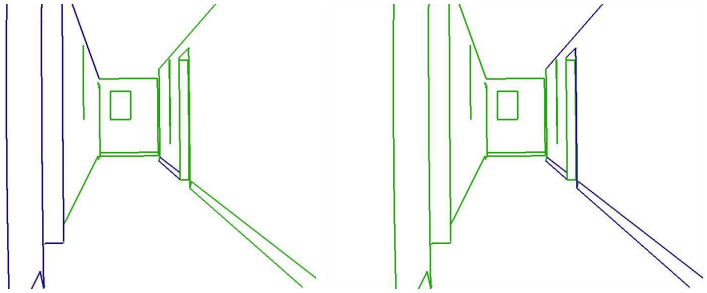
5.2 Tracking using Clusters

Having formed suitable clusters, two problems remain; hypotheses must be found for each cluster and these must then be combined to give an overall estimate of pose. It is possible to use any robust algorithm to find solution(s) within each cluster and here a simple data-driven scheme is presented which essentially performs exhaustive RANSAC. This makes tracking extremely robust to large unexpected camera motions, which is vital for the helicopter application.

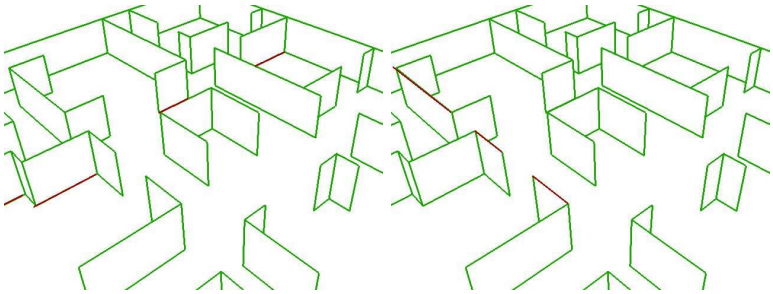
As each new frame arrives, the model is rendered at a predicted pose and its edges are clustered. Simultaneously, the image is analysed to find edge fragments (edgels). The texture change-point detector described in Section 4.3 is used to reduce the number of edgels detected in textured scenes. In Chapter 4 the



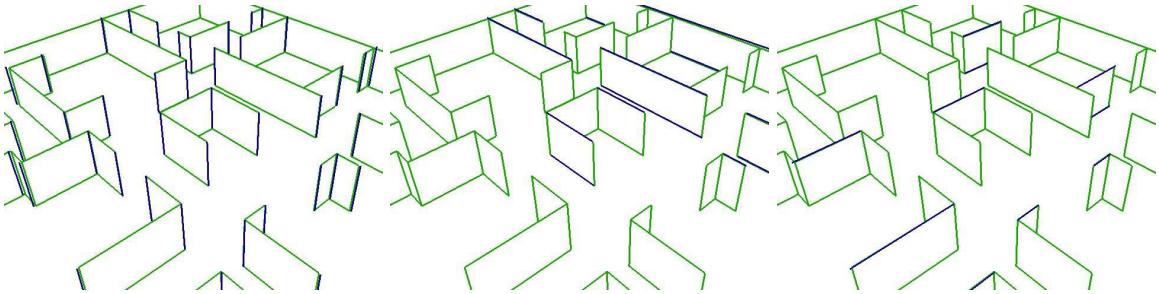
(a) 2-clusters



(b) 4-clusters



(c) 2-clusters



(d) 4-clusters

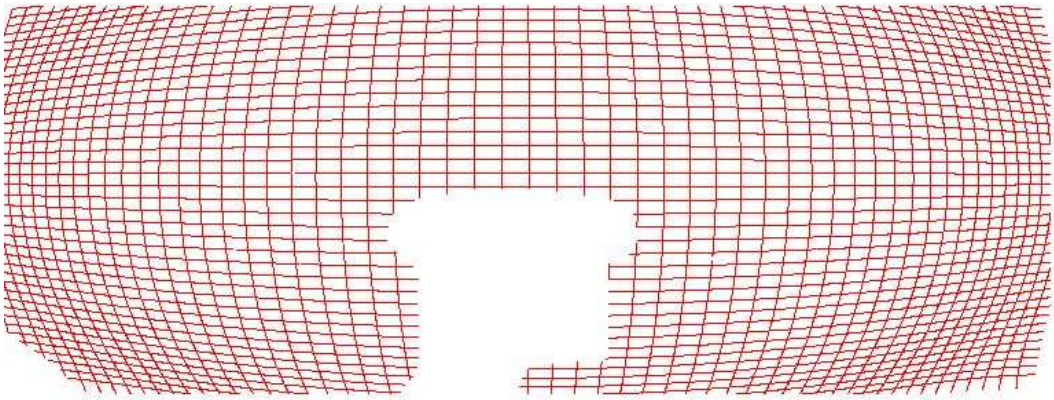
Figure 5.2: Clustering results for a corridor scene and maze scene.

change-point detector was applied locally, along lines in the image perpendicular to the model edges (see Section 3.1.3). Although this technique is computationally appealing, it has a significant drawback; because the searches for different model edges are independent, it is not possible to perform a one-to-one allocation of image lines to model edges. This means that the system described in Chapter 4 will occasionally jump to a pose where many model edges lock on to a single strong image line. Instead, if only one model edge can be allocated to a single image line, a much fairer pose score is obtained and the problem is avoided. Here therefore, the entire image is first searched to find edgels. A grid search pattern is used and on each grid line a single pass of the texture change-point detector is performed. The grid searched is a precomputed pattern which takes into account the radial distortion introduced by the lens (see Section 3.2.2). The position of detected edgels can therefore be stored in (undistorted) camera coordinates which gives the advantage that straight world edges can be easily found. Using a precomputed grid also means that a mask can be employed to prevent the detection of edgels in areas of the image which are obscured by parts of the helicopter (see Figure 5.3).

5.2.1 Tracking Within a Cluster

The maximum possible motion of the endpoint of each rendered edge segment can be computed from the maximum specified camera motion k_m. A scan of this range of the edgel-detected image finds all possible edge fragments which might match to somewhere along the model edge. Local support for each edgel is checked by simplistic edge chaining along the length of the edge which also gives an image edge direction and hence a set of possible image matches for each model edge. An improvement over the system presented in Chapter 4 is that the edgel search is extended in the direction parallel to the model edge. Although the aperture effect means that the edge will not be accurately localised in this direction, such measurements should still be included since otherwise short lines may be missed entirely.

(a) A distorted and obscured image from the camera.



(b) The grid scanned to find edgels.

Figure 5.3: Lens distortion.

For a 2-cluster c, there are just two transformation parameters to be determined and so a single match of a model edge to an image edge then gives a motion hypothesis for the cluster. A hypothesis is generated for each match of every model edge in the cluster. For a 4-cluster, all pairs of matches for all pairs of model edges are considered.

Each such hypothesised motion, h, is tested for consensus by applying the hypothesis to each edge in the cluster and searching within a short range for consensus edgels. The set of edgels, E, thus found is then used to perform linear least squares by first finding:

$$C_{c,h}^{-1} = \sum_{j \in \mathsf{E}} \hat{\boldsymbol{J}}_j^T \hat{\boldsymbol{J}}_j \quad \text{and} \quad \boldsymbol{v}_{c,h} = \sum_{j \in \mathsf{E}} \hat{\boldsymbol{J}}_j^T d_j \ , \tag{5.17}$$

where c and h index the cluster and hypothesis being considered and d_j is the perpendicular distance from the consensus edgel to the edge location. $\hat{\boldsymbol{J}}_j$ is the Jacobian of the model edge at that point (found by interpolating between the two relevant rows of J). The hypothesis pose motion vector is then refined by finding the least squares solution:

$$\boldsymbol{\mu}_{c,h} = (C_{c,h}^{-1} + \gamma I)^{-1} \ \ \boldsymbol{v}_{c,h} \ , \tag{5.18}$$

where the regularising prior γ limits motions in the null space of the cluster[1]. The cluster's motion hypotheses are sorted by residual error (with a penalty for non-consensus edgels), and all hypotheses within a threshold of the best are retained for the next stage.

5.2.2 Combining Cluster Hypotheses

Having already calculated $C_{c,h}^{-1}$ and $\boldsymbol{v}_{c,h}$ for each hypothesis h within each cluster c, combining the results from the different clusters is relatively straightforward. The covariance matrices C already embody the uncertainty of each cluster. Denoting,

[1]It is possible to include a prior which exactly prevents motions in the null space. In practice the simple prior given in Equation 5.18 works equally well.

for example, $\boldsymbol{t} = [1\ 1\ 2]^T$ to be the combination of the first hypothesis from each of the first two clusters with the second from the third cluster, the pose resulting from their combination can be found using:

$$C_t^{-1} = \sum_k C_{k,\boldsymbol{t}_k}^{-1} \quad \text{and} \quad \boldsymbol{v}_t = \sum_k \boldsymbol{v}_{k,\boldsymbol{t}_k} \tag{5.19}$$

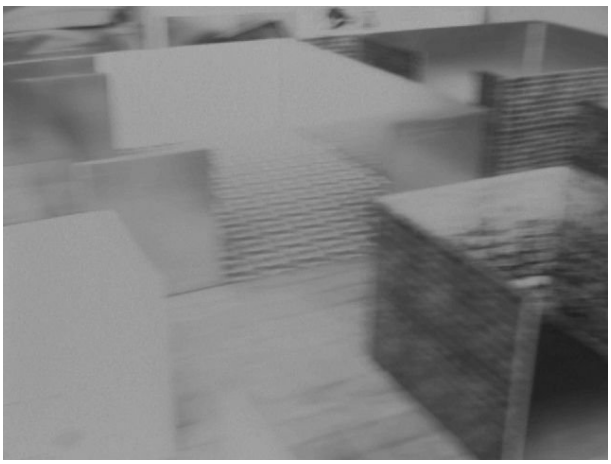
from which the pose motion can be found using Equation 5.18, with $\gamma \approx 0$. If the different clusters are independent (i.e. their search spaces $\mathcal{S}$ do not overlap) it would be expected that the residual error from the combination to be close to the sum of the residual errors from the individual cluster hypotheses. The same is true if clusters are not independent but the hypotheses agree on any of the overlapping dimensions. If the new residual error is not close, the hypotheses are not consistent with each other and the combination is invalid.

In practice, most clusters contain just one or two plausible hypotheses (those retained at the end of Section 5.2.1) and only a few clusters have much uncertainty. Hence it is computationally feasible to try all possible combinations of the different hypotheses. Further, it is also feasible to include a null hypothesis in those considered for each cluster ($C_{c,0}^{-1} = [0_{6\times6}]$ and $\boldsymbol{v}_{c,0} = \boldsymbol{0}$), which represents the possibility that non of the plausible hypotheses for that cluster are correct. The inclusion of these null hypothesess makes the system extremely robust to large motion disturbances and missed feature detections.

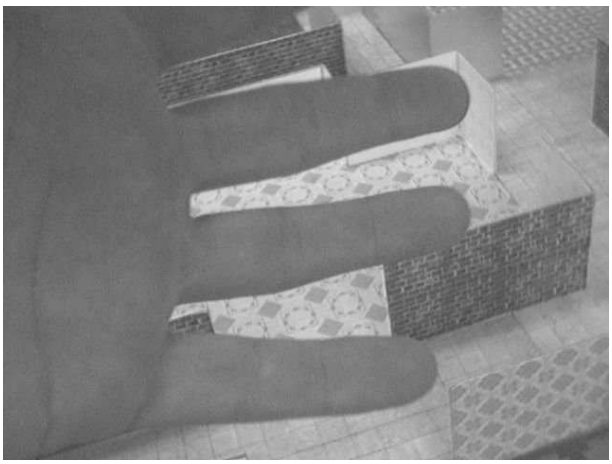
5.2.3 Tracking Results

The entire system has been tested with four different scenes. Two are the maze and the corridor scenes described in Section 5.1.7. The maze scene was filmed with a handheld firewire camera and contains blur and occlusion as shown in Figure 5.4. More importantly, motions are rapid and strongly discontinuous and hence not predicted well using a motion model; these prediction errors, which must be corrected by the tracking system, produce an edge motion error, *averaged* across the frame, in excess of 50 pixels on eight occasions during the sequence. Following such changes without making correspondence errors is a significant

challenge. A conservative estimate is that an (annealed) particle filter would need at least $5^6 = 15625$ particles to cope with such motions and hence would not operate in real-time. Frames showing the sequence being successfully tracked using the clustering system, in real-time at 30fps, are shown in Figure 5.5.



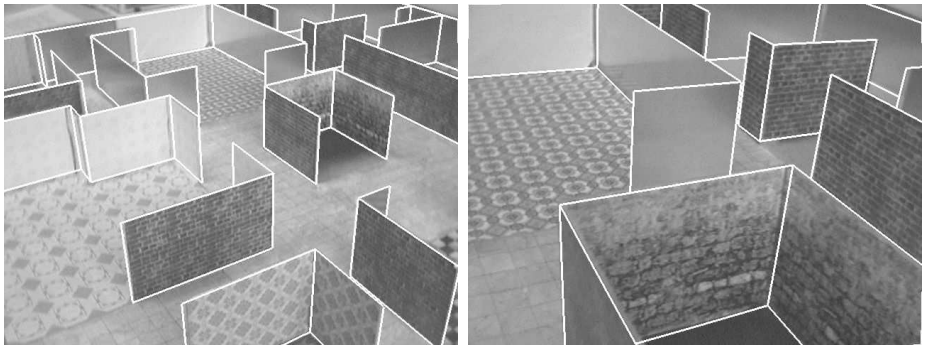
(a) Blurred images



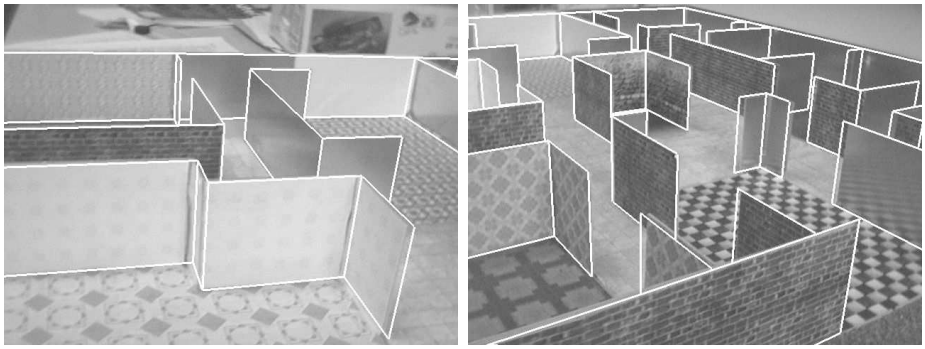
(b) Occlusion

Figure 5.4: Difficulties tackled in the maze sequence.

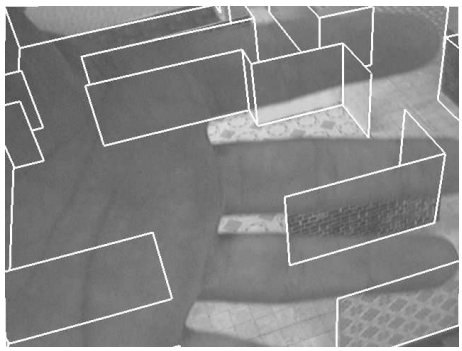
Three similar corridor sequences were filmed by the miniature transmitting video camera mounted on the helicopter, whilst the helicopter was being manually flown. Two of these sequences were tracked successfully in real-time (see Figure 5.6). The third loses track twice, once shortly before landing during three frames of interference and again due to very large motions during landing.



(a) Frame 20 (b) Frame 60



(c) Frame 80 (d) Frame 250

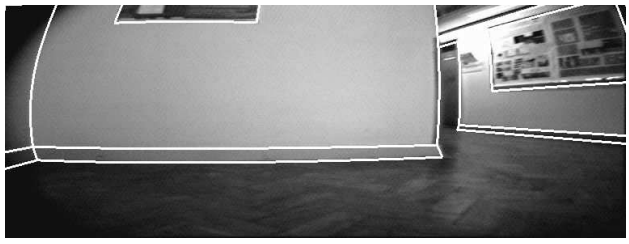


(e) Frame 400

Figure 5.5: The clustering system tracking the maze sequence correctly.

(a) Frame 1

(b) Frame 500



(c) Frame 709

(d) Frame 1000

(e) Frame 1250

Figure 5.6: The clustering system tracking the corridor sequence correctly.

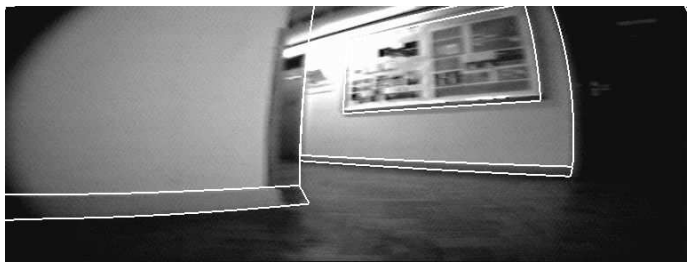
As before, the sequences were also tracked with two other systems, a demonstration version of 'boujou' and an earlier edge based system [Drummond & Cipolla 1999]. Table 5.1 shows the number of tracking failures for each. 'boujou' is a commercial product (`www.2d3.com`) which performs offline bundle adjustment of tracked interest points. It performed reasonably on the maze sequence, failing only when the hand occluded large parts of the image (demonstrating the advantage of having a model), or due to the repetitive textures. Unfortunately the very high level of image noise and blur for the corridor sequences meant that the longest tracked section was only a few frames long. Of course, this is not a particularly fair test since boujou has no prior model and is not particularly optimised to work with poor quality video; however, it demonstrates the difficulty of solving the problem using a point-based tracking scheme.

Table 5.1: Results from tests on four sequences, showing the number of tracking failures for the three different schemes.

		Number of Failures		
Seq.	Frames	boujou	Earlier Edge	Clustering
Maze	577	18	108	0
Corr1	1311	> 100	30	0
Corr2	1101	> 100	23	0
Corr3	1080	> 100	15	2

To demonstrate the flexibility of the clustering approach, the system has been extended to work with articulated models. The first is the extremely simple eight-dof torso model shown in Figure 5.8. Performance is limited and it is not intended to compete with state of the art real-time systems: the example is included to demonstrate that the clustering makes sensible simplifications. For example, though many motion parameters affect the position of the lower arm, it will always appear as two parallel image edges which then form a 2-cluster. The system is consequently able to disambiguate the arms after they occlude each other, despite not using a motion model. The feature clusters automatically generated for the torso model are generally constant and bear strong similarities to the partitions which would be used in Partitioned Sampling [MacCormick &

(a) Frame 500



(b) Frame 709

Figure 5.7: The earlier edge-based system falls into incorrect local minima (cf. Figure 5.6)

Isard 2000]. However, a hierarchical search is typically used with partitioning; here the searches are independent and hence there is no possibility of errors propagating down the hierarchy.

As mentioned in Section 2.3, the method described for the Annealed Particle Filter in Deutscher *et al.* [2001] automatically forms hierarchical partitions of the search space when all measurements agree on parameter values. A crossover term is also employed to allow the splitting of the search space into non-overlapping parallel partitions. Here, clusters of features are used, rather than partitions of the whole search space. These clusters are independent and can thus be searched in parallel rather than in a hierarchy. Additionally, unlike parallel partitions, they can overlap and hence can be generated more freely. Finally, in many situations such as the corridor or maze scenes, there are few or no parameter directions which can be independently partitioned for *all* the measurements. Clustering allows subsets of measurements to be considered and hence works well in such situations.

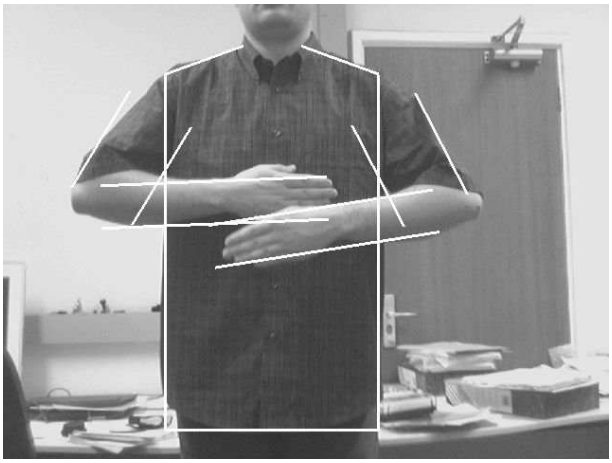


Figure 5.8: A simple real-time torso tracking system based on clustering is able to distinguish the arms correctly after they cross, despite not using a motion model.

A further example demonstrates nine-dof (locally eight-dof) tracking of a radio controlled tank with a moving camera. In this case dynamic clustering provides significant advantages over static partitions. As for the corridor and maze examples, the structure of the scene at each particular pose is exploited to divide the search space. Further, since the tank cannot move sideways, the edges of the tank can often be included in clusters which largely contain scene edges. Hence the tank can often aid the tracking of the surroundings. A sample frame is shown in Figure 5.9.

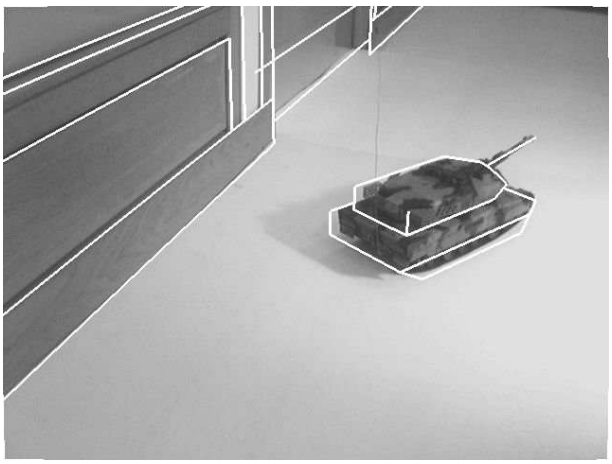


Figure 5.9: A radio controlled tank is tracked whilst the camera is also free to move.

5.3 Conclusions

This chapter has presented a method for automatically clustering measurements based on their Jacobians. This clustering provides three benefits over traditional partitioning. Firstly the system is automatic and does not require a human to define the partitions. Secondly the clustering is dynamically updated as the problem changes. In the case of tracking this means that the clusters can benefit from properties of the particular pose rather than needing to be valid for all poses. Lastly, the clusters are independent of each other and hence can be searched in parallel before combining their results. This means that there is no possibility of error propagation, as is possible in hierarchical schemes.

Most examples showed the application of clustering to straight edge tracking. However, it is believed that the feature clustering technique is general and so can also be usefully applied to other vision problems such as point tracking or the tracking of implicit surface models.

For the automatic control of the MAV, the pose of a camera must be tracked as it moves past known polyhedral objects or surroundings. To ensure the tracking system is tolerant of poor image quality and large motion disturbances, a data driven approach has been employed. This considers a wide variety of possible matches between image and model features. To ensure this process can be performed in real-time, the new dynamic clustering method has been used to significantly reduce the combinations of matches which must be considered. The resulting system has been demonstrated on a variety of tracking scenarios and shows a significant improvement in motion robustness over previous techniques. In particular, reliablity has improved to the point where the significant majority of helicopter flights are successfully tracked and the automatic control of the MAV would appear to be viable.

Chapter 6

Likelihood Filtering

The visual tracking system described in the previous chapter allows the pose of the MAV to be robustly determined during flight. However, to damp away violent oscillations in the MAV's motion, it is also vital that a reasonable estimate of its velocity is provided. Also, when the motion of the MAV is unpredictably disturbed (for example by a wind gust) it is important that such sudden velocity changes are reported with minimal lag so that the controller can respond rapidly.

Some attempts have been made to measure velocity directly from the image by examining blur [Klein & Drummond 2005; Rekleitis 1996]. Unfortunately, the results are inaccurate or not obtainable in real-time and extensions to providing full six dof velocity are likely to be challenging. Hence it is necessary to treat velocity as a hidden state and attempt to deduce it from the tracked pose information.

The underlying strength of the tracking system is that it is able to directly find peaks of the measurement likelihood function. Unfortunately, due to the significant image noise and modelling errors, the peak of the likelihood function is generally a very noisy estimate of the MAV pose. Obtaining velocity simply by differentiating the pose estimates, using finite differences between consecutive frames, produces extremely poor results.

6.1 Kalman Filtering

One potential method for obtaining velocity estimates would be to process the likelihood measurements using an EKF (see Section 2.6). Fundamentally, the Kalman filter provides a trade-off between believing the prior prediction and believing new measurements. In the case of a 12 dof 'constant velocity' filter, which models acceleration as a white noise process, the 'ratio' of the process noise covariance to the measurement covariance controls the belief in zero acceleration versus the pose given by the peak of the likelihood function.

Ideally, both the measurement covariance and the process noise covariance would be accurately measured. In practice, only approximate estimates of the scaling of the measurement covariance relative to the process noise are available and this ratio is often assumed to be constant and is tweaked to obtain the best performance. When the ratio is set so as to favour measurements, actual noise on the measurements is propagated and hence the velocity estimates are noisy. As the ratio is adjusted in favour of the prior (lower process noise) the velocity measurements become smoother, but the filter reacts slowly to accept changes in velocity. In many previous approaches this is associated with the system losing track, since the filter is often used to provide the prior pose indication to the tracking system. Here, track will not be lost since it is possible for the tracking system described in the previous chapter to run independently from the Kalman filter. However, if the ratio of process to measurement noise is adjusted to obtain sufficiently smooth velocity estimates, the estimates provided by the filter begin to lag behind the true state by an amount which is unacceptable for the real-time control of the MAV.

Perhaps the best explanation for the Kalman filter failing to perform adequately is that the distribution of accelerations experienced by the MAV is far from Gaussian. Figure 6.1 shows the distribution of lateral accelerations obtained during a test flight, as measured by the external tracking system described in Section 6.4 (which can be treated as ground truth). Although the central portion of the measured distribution is roughly Gaussian shaped, the tails of the distribution

are substantially higher than those of a Gaussian distribution. These 'high tails' are caused by large, unpredictable forces such as gusts and the ground effect. A Kalman filter which is tuned to model the central peak cannot respond sufficiently rapidly to these effects, whereas a filter which is tuned to best cover the entire distribution gives poor velocity estimates when the accelerations are within the central peak.

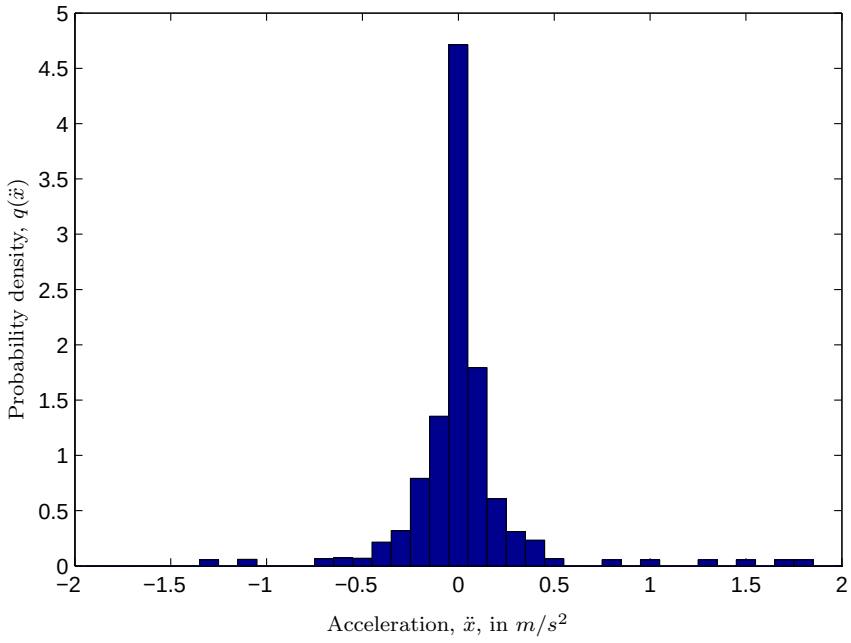


Figure 6.1: The distribution of accelerations obtained during a typical flight of the MAV.

6.1.1 Accuracy versus Reaction Speed

Even if a constant velocity model is not strictly applicable, in many cases a Kalman filter can be used successfully. For the MAV application, the problem is the large noise present in the pose estimates from the visual tracking system. Consider a typical situation where the MAV has remained stationary followed by a measurement suggesting that it has moved 2cm to one side. The question is whether to attribute this 2cm to noise, to assume that the MAV has actually

suddenly started to move to the side, or to choose an estimate somewhere between the two.

The solution to this problem is a trade-off between the required accuracy and the speed at which the filter responds. As described in Section 3.1.1, the tracking system requires a prediction of the camera pose for an new incoming image. It is not necessary for the prediction to be particularly accurate, given the tracking system of Chapter 5 which can tolerate large errors. However, it is necessary for its speed of response to be high; otherwise, if the MAV has indeed suddenly started to move to the side, the prediction error will rapidly become too large. For predicting motions it is therefore appropriate to use a Kalman filter with high process noise so that it responds rapidly to motion changes. The inability of these fast-reacting filters to provide accurate pose predictions explains why unexpectedly large motion constants are needed in Section 5.1.7.

For accurate velocity estimates however, a filter which has a slower response is necessary. The chances are that the 2cm measurement is due to noise, and reporting it as a large velocity change will inevitably lead to noisy velocity estimates. The required smoothing can be obtained using a Kalman filter with a low process noise. However, if in the future it becomes apparent that the MAV did actually experience a large disturbance, the Kalman filter will be unable to account for such a large acceleration. Although it will eventually converge on the correct state, there will be a significant time lag and this lag is unacceptable from a control point of view, particularly since it is exactly these unpredictable motion disturbances that require a rapid response from the controller. Although some lag is inevitable, it can be significantly reduced by allowing the filter to simultaneously consider these different hypotheses so that the new mode can be reported if significant evidence of a large motion disturbance becomes available. This is demonstrated next with a didactic example.

6.1.2 Didactic Example

Figure 6.2 shows an attempt to track a simple 1D path using a Gaussian acceleration approximation. Using a narrow process noise results in the accurate recovery of most parts of the path but, as expected, the filter is unable to recover successfully the sharp change in velocity. If a Kalman filter with a wider process noise is used, the velocity change is somewhat more rapidly followed but at the cost of increased belief in the measurements. This leads to the measurement noise now causing significant noise on the velocity estimate. Adding even more process noise results in very poor velocity estimates with only a slight improvement in the speed of response.

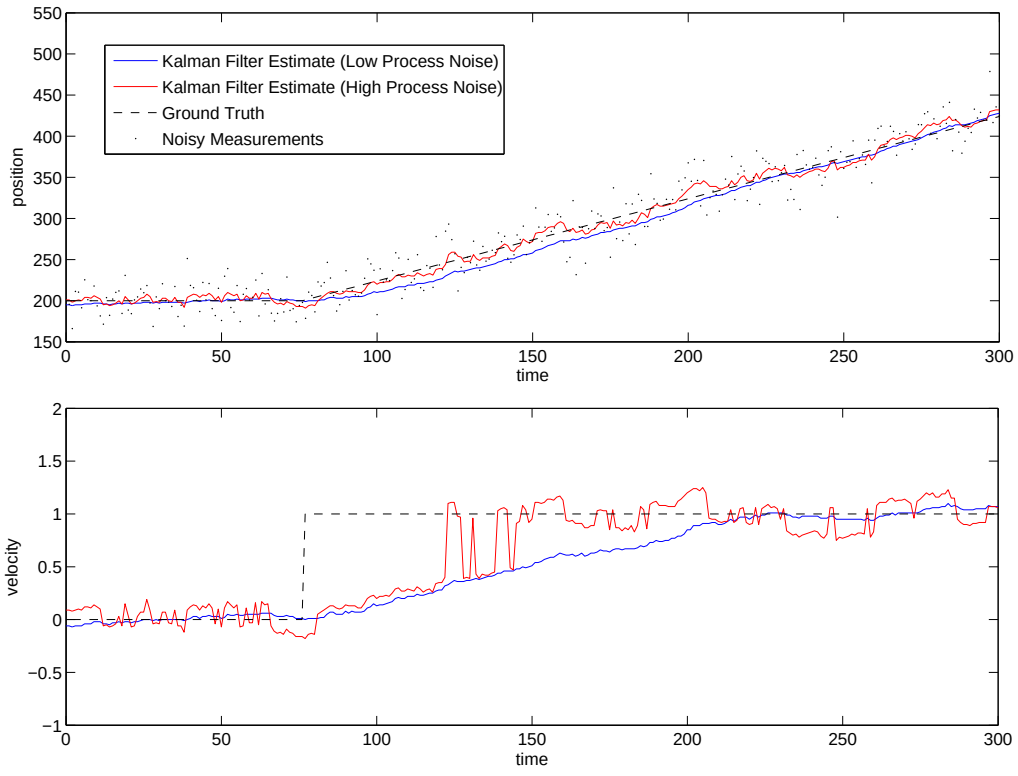


Figure 6.2: The Kalman filter with a Gaussian acceleration distribution finds a poor approximation of the true velocity from the noisy measurements if the accelerations are sufficiently non-Gaussian.

To obtain good velocity estimates, it is therefore necessary to represent the accel-

eration distribution more accurately, for example using a Gaussian-plus-uniform distribution. Implementing such a filter using dense sampling (just for this didactic example) gives the results shown in Figure 6.3. When this Gaussian-plus-uniform process noise is added to the filter at each time step, it blurs out the peaks as usual, but now the broad tails contribute to give a finite and approximately uniform 'sea' between the peaks. Now the inclusion of each single measurement into the filter has two effects on the posterior. As usual, if the measurement is close to a peak of the filter it will reinforce the strength of that peak in the posterior. However, the likelihood measurement will also combine with the uniform area to give a peak in the posterior very close to the measurement. This double effect of a single measurement leads to multiple modes in the posterior. Hence soon after the sudden velocity change, two prominent modes form (as shown in Figure 6.4) and, once the new peak becomes higher, the filter switches to the new position and velocity, with only a short time lag.

6.2 Theoretical Representation

It is relatively straightforward to mathematically describe a filter which correctly maintains a multi-modal posterior estimate and which is able to use a non-Gaussian acceleration distribution.

Assuming, for now, that the parameter space is linear, if $\boldsymbol{\mu}$ is a vector of pose parameters and $\dot{\boldsymbol{\mu}}$ the vector of corresponding velocities then, using Bayes' rule, the posterior probability of the states at time t, given the data up to that time $D_{0..t}$ is:

$$p(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}|D_{0..t}) \quad \propto \quad p(D_t|\boldsymbol{\mu})\ p_t(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}) \ . \tag{6.1}$$

Here $p(D_t|\boldsymbol{\mu})$ is the likelihood function used by the tracking system. The prior probability $p_t(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}})$ can be found by marginalising the posterior found for the previous time step $t - \delta t$ over all possible velocities $\dot{\boldsymbol{\alpha}}$ (the Chapman-Kolmogorov equation):

$$p_t(\boldsymbol{\mu}, \dot{\boldsymbol{\mu}}) = \int_{-\infty}^{\infty} p(\boldsymbol{\mu} - \dot{\boldsymbol{\mu}}\delta t, \dot{\boldsymbol{\alpha}}|D_{0..t-\delta t})\ q(\dot{\boldsymbol{\mu}} - \dot{\boldsymbol{\alpha}})\ d\dot{\boldsymbol{\alpha}} \ , \tag{6.2}$$

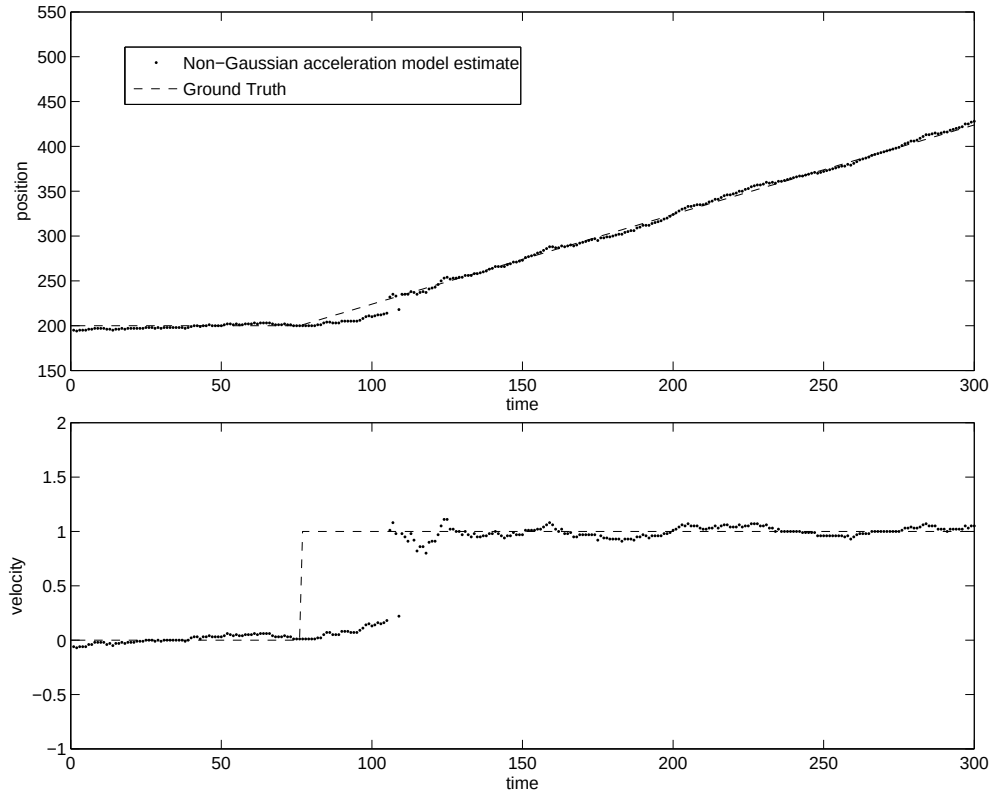
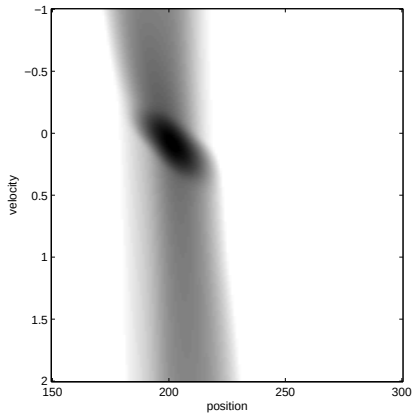
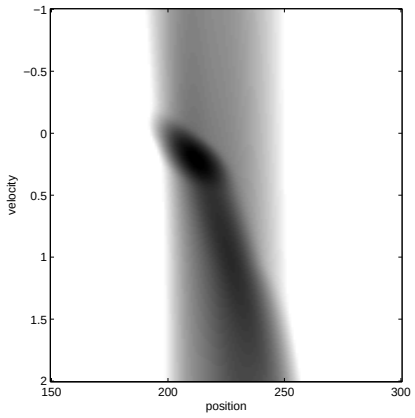


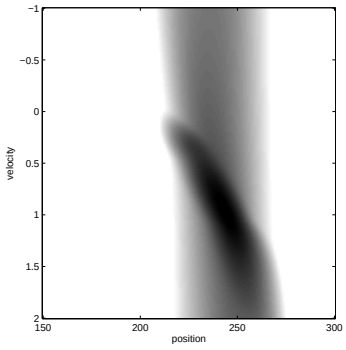
Figure 6.3: For measurement data as for Figure 6.2, using an accurate acceleration model means that the velocities are much more accurately recovered, even in the presence of significant measurement noise.



(a) Time=80



(b) Time=100



(c) Time=120

Figure 6.4: The posterior probability density function for various times, showing the new velocity mode develop.

where $q(\dot{\boldsymbol{\mu}} - \dot{\boldsymbol{\alpha}})$ describes the acceleration probability that the velocity changed from $\dot{\boldsymbol{\alpha}}$ to $\dot{\boldsymbol{\mu}}$ at the start of the last time interval.

With a real pose estimation problem, the six degree of freedom parameter space is non-linear. Hence, just as with the extended version of the Kalman filter, it is necessary to linearise the problem about the current pose. However, unlike the Kalman filter, $p_t()$, $p()$ and $q()$ in Equation 6.1 are not restricted to being Gaussian distributions, but instead are general multi-modal distributions.

Recently, and with particular reference to tracking systems, such multi-modal distributions have often been described using a representative set of weighted particles. Such particle filtering provides one method for implementing the system described by Equation 6.1. However, filtering the MAV's pose in this way incurs two large computational costs. Firstly, the inclusion of velocity in the state vector creates a 12 dof distribution and this means that the number of particles needed to successfully represent the distribution is massively increased. Secondly, in order to represent the significant 'width' of the acceleration model (i.e. $q()$ in Equation 6.2 is non-zero over a significant range), each strong particle would have to be re-sampled many times. Such computational increases mean that these previous approaches are not practical in a real-time MAV system. Instead, a real-time solution can be achieved by directly filtering peaks of the likelihood distribution.

6.3 Likelihood Filtering

Conceptually, the likelihood filter simultaneously considers several hypotheses for trajectories through the parameter space. Each is stored as the pose and velocity at the trajectory's endpoint, using a 12 dof Gaussian distribution to represent the mean and uncertainty of the endpoint. As a new frame arrives, the mean of each endpoint pose is projected forward using its mean velocity, and process noise is added to its covariance to give a prior hypothesis pose for that frame. The tracking system operates independently of these priors and returns a list of

likelihood peaks. The list is searched to find the 'best match' for each hypothesis and this new Gaussian measurement is combined with the prior hypothesis to give a Gaussian describing the posterior distribution for that trajectory. Figure 6.5 shows the stages of this process for a simple example.

To implement Equation 6.1 correctly, at each time step, each trajectory would need to be split into many new trajectories, one for each of the new likelihood peaks. Such 'mode explosion' is avoided by assuming that the central peak of the acceleration distribution $q()$ is relatively narrow. This means that one branch of a trajectory will significantly dominate the others splitting from the same trajectory. Hence, rather than split the trajectories, the branch with the greatest posterior probability (the 'best match') is retained as the continuation of that trajectory. This same assumption is made by most single-mode tracking schemes and by Cham & Rehg [1999].

However, as discussed in Section 6.1, it is important for this application to consider an acceleration distribution which is not just a narrow Gaussian but a narrow Gaussian plus uniform distribution. This means that the prior distribution formed from the trajectory endpoints is a collection of narrow Gaussian peaks separated by a finite sea. Hence as well as combining likelihood measurements with trajectory endpoints, a new trajectory must also be formed at each likelihood peak. These new trajectories will have very uncertain velocity estimates since they are formed under the assumption that a large acceleration has just occurred.

The assumption that the acceleration distribution is composed of a rather narrow Gaussian plus an essentially uniform distribution reduces the problem to a linear increase in the number of trajectories with time. Old and unlikely trajectories must therefore be culled to keep the number of trajectories constant.

Mathematically, the likelihood and posterior at time t are represented as Gaussian mixtures models with L 6 dof and M 12 dof mixtures respectively. A specific posterior Gaussian m represents the hypothesis that the last large acceleration

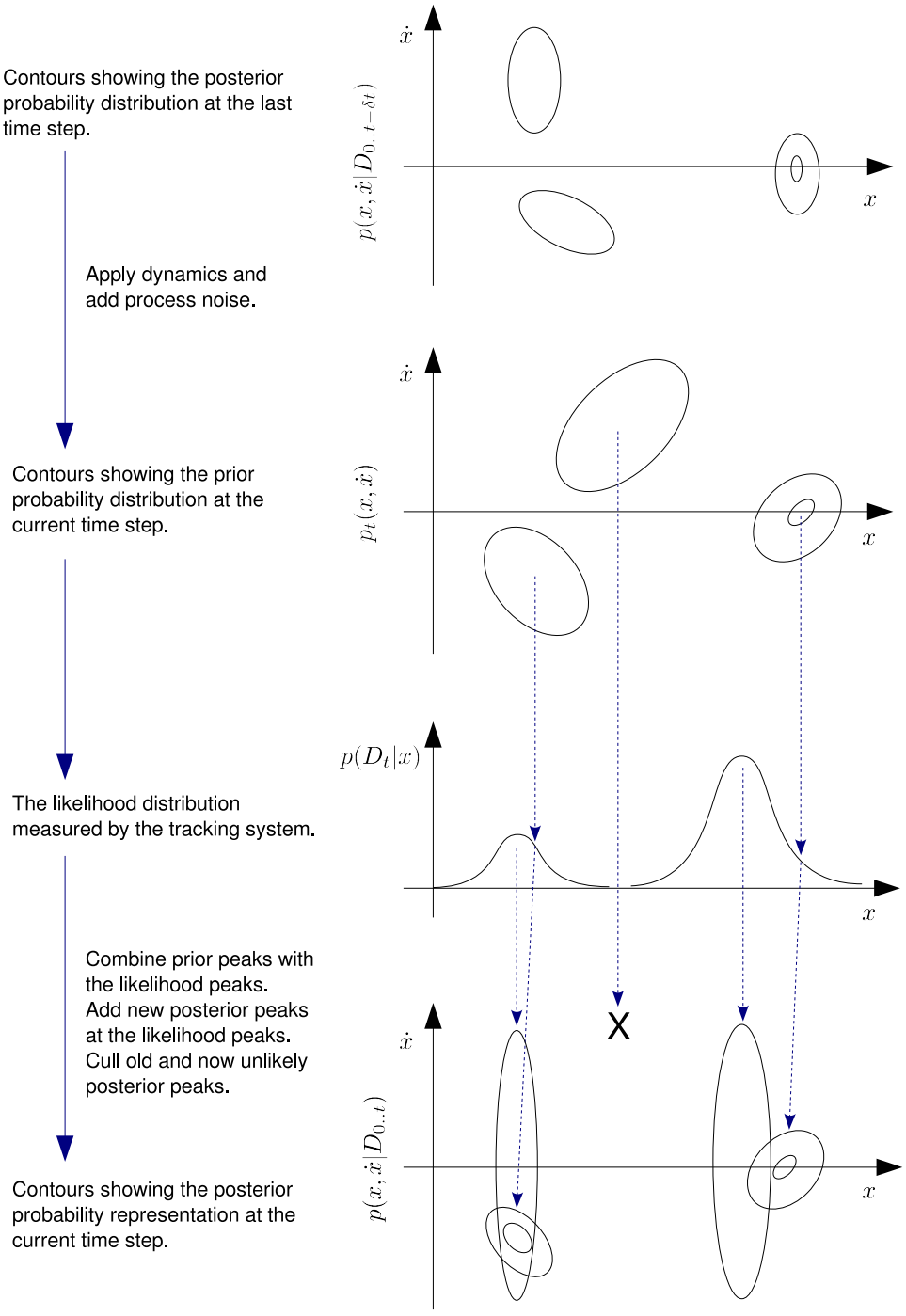


Figure 6.5: Filtering the likelihood hypotheses.

occurred at a certain age $t - \Delta T_m$. Due to the large acceleration, the likelihood distributions before this age have no effect; at $t - \Delta T_m$, m was therefore a Gaussian with pose centred on one of the likelihood mixture means with a very uncertain velocity. For this hypothesis, only small accelerations have occurred since $t - \Delta T_m$ and only the best likelihood peak at each successive time since will have contributed significantly. Hence at time t the posterior is formed of many groups of L mixtures, each group representing the hypothesis that the last large acceleration occurred at a certain age.

6.4 Filtering Results

Figure 6.7 shows plots of the MAV's horizontal velocity during a typical test flight. The ground truth measurements were obtained by mounting two high power infra-red LEDs on the MAV and observing them with a tripod-mounted video camera. An infra-red filter was used to filter out all light apart from that emitted by the LEDs, so that images similar to that shown in Figure 6.6 were obtained. A telephoto lens was employed with the camera mounted at a distance of 8m from the helicopter's test flight space and the LEDs were positioned on the MAV's pitch axis. Its height, lateral position and roll could then be accurately determined from a single image, since yaw, pitch and longitudinal position will have a very small effect on the observed LED position. A simple thresholding and flood-filling algorithm was used to measure the 'mass centre' of each LED in the camera image and the helicopter's height, lateral position and roll was determined from the two positions using a simple calibration. Since the LED positions are relatively noiseless, accurate 'ground-truth' estimates for the helicopter's velocities can be found using finite differences and a non-causal smoothing filter.

Overlaid on Figure 6.7 are the estimates of velocity obtained when using a Kalman filter to process the noisy pose estimates from the visual tracking system. With a process noise variance corresponding to $1m/s^2$ the filter response is rapid but the velocity estimates are very noisy. With values around $0.5m/s^2$ the velocity estimates are still too noisy but now a lag in the response is visible. With $\sigma =$

(a) Plan view of the setup used.

(b) A typical image obtained from the ground truth camera.

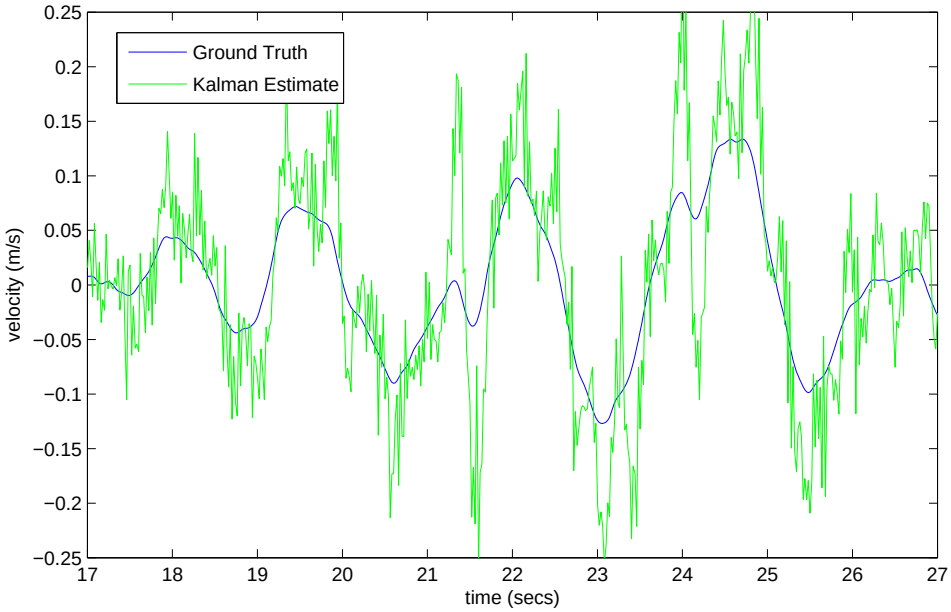
Figure 6.6: Obtaining ground truth measurements for the pose of the heliopter.

$0.2m/s^2$ the velocity estimates are improved to an acceptable level, but the filter response now lags behind the truth by around 0.2 seconds, which causes serious problems for closed loop stability.

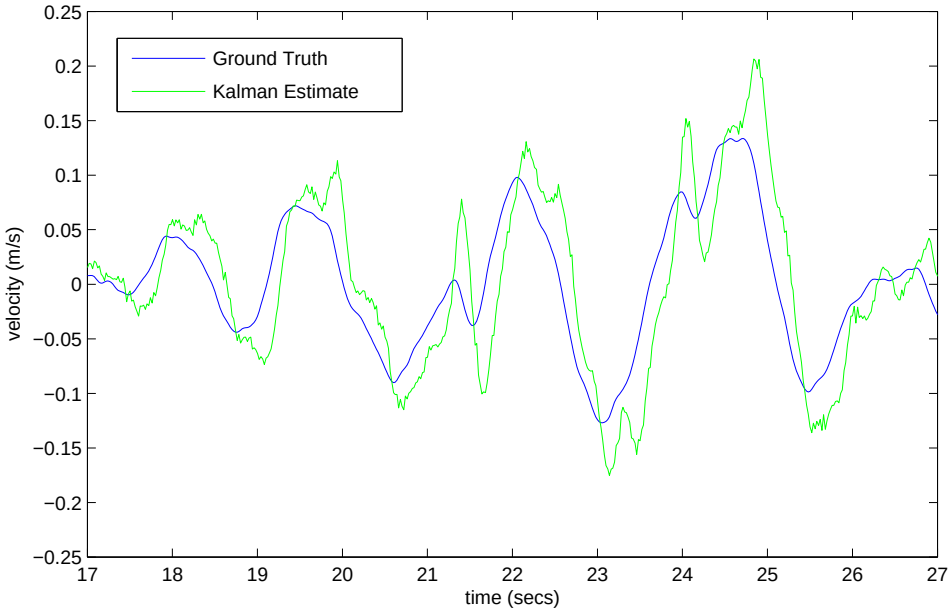
Figure 6.8 shows the velocity estimate obtained using the multi-modal filter just described. The central peak of the acceleration model used is equivalent to Kalman filtering with $\sigma = 0.2m/s^2$ and the noise on the velocity estimates is similar to that seen in Figure 6.7b. However, there is considerably less overshoot visible at times of rapid acceleration, since these situations are now modelled. More importantly, the time lag at velocity changes is now significantly reduced. The result is an accurate and yet responsive velocity estimate which can be used by the MAV controller.

6.5 Conclusions

This chapter discussed the trade-off between latency versus noise propagation when filtering pose estimates to obtain velocity estimates. For a control system to successfully fly the MAV, it must have accurate velocity estimates. Moreover, if a large acceleration occurs, it is vital that the filter reports the new velocity quickly since it is exactly these unpredictable motion disturbances that require a rapid response from the controller. Although the Kalman filter can often be applied in situations where the underlying assumption of white process noise is not strictly true, doing so means that its results suffer from significant time delays. Instead, by employing a better process noise model and hence a multi-modal filter, these time delays can be reduced. The multi-modal filter is also able to systematically handle the multi-modal likelihood distribution obtained from the tracking system described in Chapter 5.



(a) Kalman velocity noise of $\sigma = 1m/s^2$



(b) Kalman velocity noise of $\sigma = 0.2m/s^2$

Figure 6.7: The MAV lateral velocity during a test flight and the Kalman filter estimate.

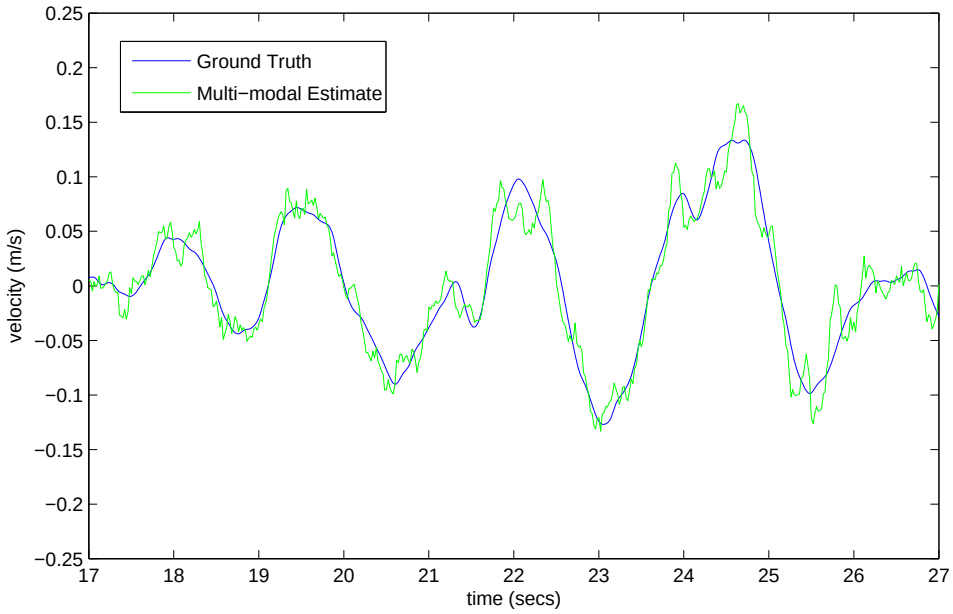


Figure 6.8: The MAV lateral velocity during a test flight and the velocity estimated using the multi-modal filter.

Chapter 7

Automatic MAV Flight

Having obtained robust and accurate estimates of the MAV's pose and velocity, its position can now be automatically controlled by calculating the required rotor speeds and sending these back to the helicopter using a custom made USB interface (Appendix B) and standard radio control transmitter. This chapter describes the development of the complete system and presents results from a number of test flights.

7.1 Real-Time Processing

To ensure that visual track is not lost and closed-loop stability is maintained, it is important that the tracking algorithm described in Chapter 5, the filtering described in Chapter 6 and the control calculations described in the following section can all be performed within the 20ms of a single video field. The tracking and filtering algorithms are highly parallelisable, as shown in Figure 7.1, and so use is made of the two processors available in the controlling PC (see Table 1.2 for specifications).

A simple architecture was used to enable the easy development of a multi-threaded application. A single queue of 'command' objects is initialised at startup and

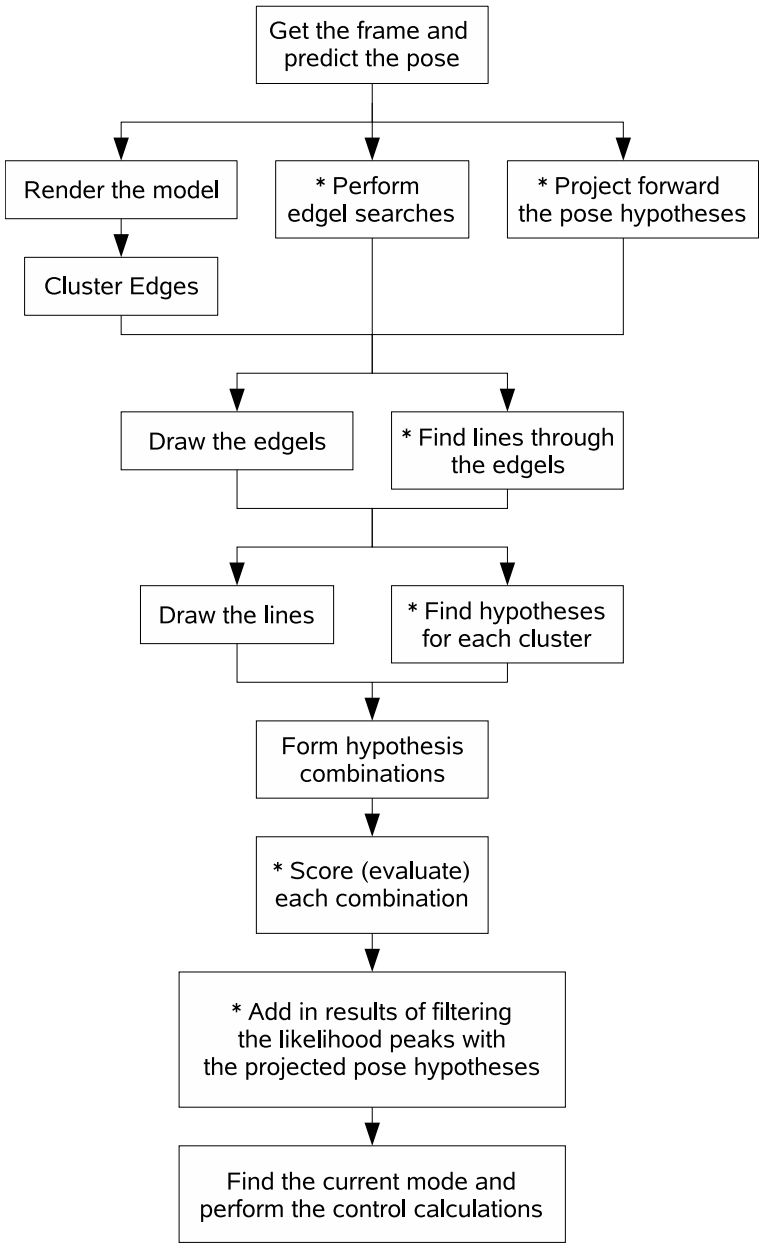


Figure 7.1: The operations performed for the processing of each frame. A '*' next to an operation indicates that many separate instances of this operation can be performed in parallel, together with any other operations in that row.

access to this queue object is protected by a mutex. Also at startup, one or more slave threads are created which sleep until signalled by the enqueuing of a command. The thread then dequeues the command, executes it and returns any results to member variables of the command object. Some commands require a large block of memory to store flags or temporary results. One such block is allocated to each thread so that commands can use this 'thread local' storage without needing to obtain a lock.

The master thread is used to control synchronisation of the commands. A typical section of code looks like:

```
for(int ii=0;ii<sz;ii++) exec.enqueue(&cFindLines[ii]);
exec.masterExec(masterData, &cEdgelsDraw);
exec.masterRun(masterData);
```

The first line adds many 'find line' command objects to the queue operated by the `exec` object. This causes the slave thread to be woken and it immediately begins execution of these new commands. The master thread meanwhile executes the 'draw edgels' command[1]. Once this is completed, the third line instructs the master thread to aid the slave by executing commands from the queue. The `masterRun` procedure does not return until the queue is empty and all commands have been completed. Hence following code can safely assume that the 'line' data structures are all filled and can begin (see Figure 7.1) to add 'cluster hypothesis finding' commands to the queue.

For a typical set of tracking parameters and one pre-recorded test sequence, the average field processing time when using only a single thread (and single processor) was measured as 20.2ms. Using two threads (with two processors) and the architecture described above reduces the average processing time to 11.7ms, which demonstrates the highly parallelisable nature of the tracking and filtering system. For interest, a comparison was also made on a machine with a single

[1]OpenGL provides no thread safety and hence all OpenGL commands must be executed by the master thread.

2.8GHz hyperthreaded processor. Here, the use of two threads rather than one reduced the computation time from 36ms to 30ms. Using more than two threads generated no significant reduction on either machine.

7.2 MAV Controller Design

The four-rotor helicopter is an under-actuated, non-holonomic device. It can be flown directly to any position x, y, z in space and its yaw angle ψ can be independently controlled. Its roll ϕ and pitch θ directly determine its lateral acceleration and hence these parameters cannot be independently controlled. The MAV is designed for manual control and hence the four radio channels do not directly control the individual motor speeds. On the helicopter is a small control circuit which adjusts the motor speeds with the radio demands and signals from onboard miniature gyroscopes. First order approximations of the channel allocation is given in Table 7.1.

Channel	Controls	Approximate Linearisation
1	Roll	$u_1 \sim 0.25\dot{\phi}\ \mu s$
2	Pitch	$u_2 \sim 0.25\dot{\theta}\ \mu s$
3	Height	$u_3 \sim 0.15\ddot{z}\ \mu s$
4	Yaw	$u_4 \sim 0.25\dot{\psi}\ \mu s$

The linearisation is given in terms of the radio controller pulse length (see Appendix B) where the range is 0.7ms.

Table 7.1: MAV radio channel allocation.

The linearisation given for height is applicable when the MAV is flying above the ground effect; at lower heights the value of u_3 starts to control the vertical position rather than acceleration. The control values $u_{1\ldots4}$ are each subject to a 'trim' offset. These constants are measured by the USB interface when the software is started and applied before the demands are sent out to the radio controller.

7.2.1 Time Lag

The open loop contains a significant time delay between a demand being sent to the MAV and its effect being observed by the visual system. Measurements suggest that the delay is of the order of 100ms. At least 60ms of this is due to the camera integration, image capture and tracking. The filtering described in Chapter 6 and mechanical delays account for the remainder. This order of delay has a serious implication for closed loop stability. In part, its effect can be reduced through the use of a forward prediction model. This uses the last 100ms of demands sent out to the helicopter in addition to the most recently available pose estimate to obtain a prediction of the current state. The following state update equations are used for each 20ms field:

$$\begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_{t+0.02} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0.02 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.02 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0.02 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.2 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.2 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix}_t + \frac{0.02}{500} \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} \tag{7.1}$$

No attempt is made to predict vertical speed based on the value of u_3, as explained in Section 7.2.5.

These state predictions do not completely remove the effect of the time delay; errors can still be caused by variations in the delay or by inaccuracies in the prediction due to modelling errors or variations. Such errors can be viewed as disturbances and must be tolerated by the control margins.

7.2.2 Yaw Control

The yaw of the helicopter is the least critical of the controls required since it has no further effect on the helicopter's motion. Using the interface described in Appendix B meant that the control loop for yaw could be independently developed and tested, whilst the the remaining channels were manually controlled. Figure 7.2 shows the closed loop for yaw. At low frequencies it was anticipated that the on-board controller produces $\dot{\psi} \propto u_4$ and that the vision system transfer function is purely a delay.

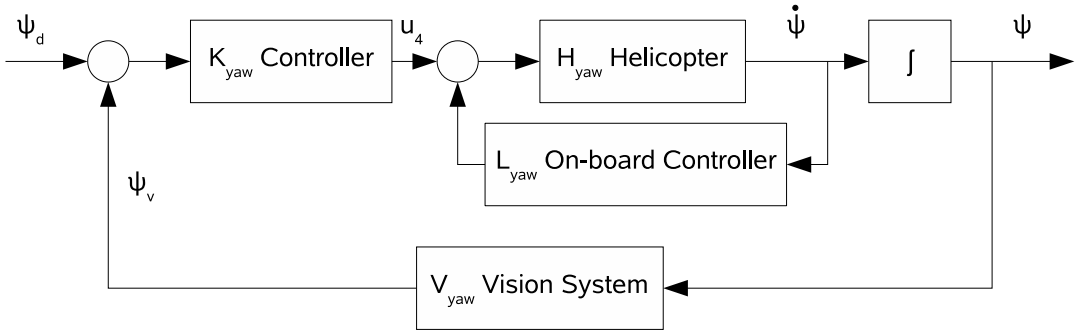


Figure 7.2: The control loop for MAV yaw (ψ).

The frequency response of the open loop system from u_4 to ψ_v was measured and is shown in Figure 7.3. The magnitude response is lowpass, with a flat response up to a -3dB point at approximately 2Hz. The phase plot shows a phase lag which increases linearly with increasing frequency. The gradient of this line corresponds to a system time lag of approximately 100ms.

In a typical flight, the demand for yaw will be largely constant and hence, since disturbances tend to have a relatively small effect on yaw, only low bandwidth yaw control is required. Further, small steady state errors are normally acceptable since (unlike roll and pitch) any errors will have no further effect on the helicopter's motion. Hence a very simple proportional controller ($K_{yaw} = 2000$) is entirely satisfactory and this was verified through many test flights.

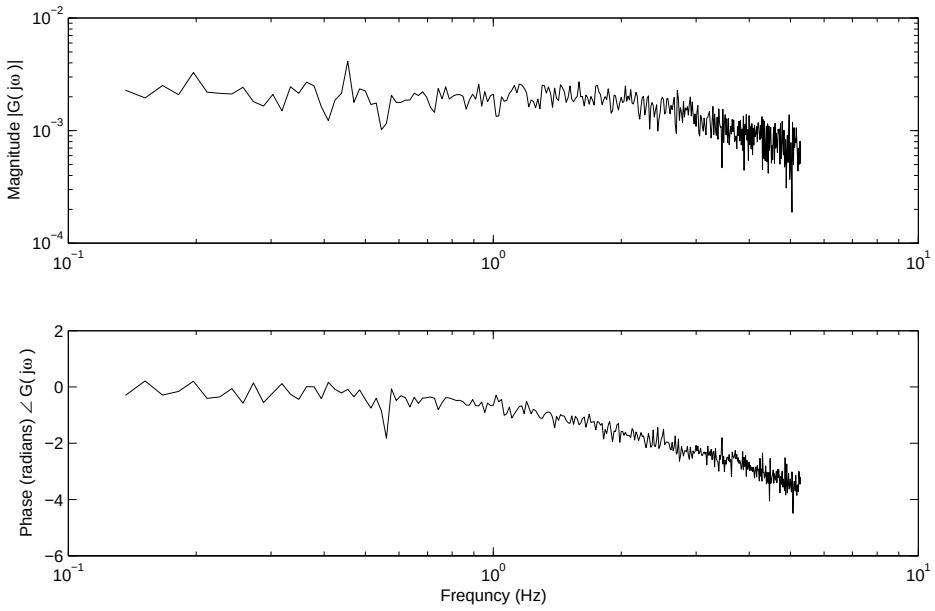


Figure 7.3: The frequency response of the open loop system from u_4 to ψ_v.

7.2.3 Roll and Pitch Control

Given the symmetry of the MAV, it is expected that control of its roll and pitch should be independent and that their responses should be the same. Hence from here on only roll is considered and its closed loop is shown in Figure 7.4. Unlike yaw, much higher bandwidth control is required for roll, since its value has a direct relationship to lateral acceleration, $\ddot{x}$.

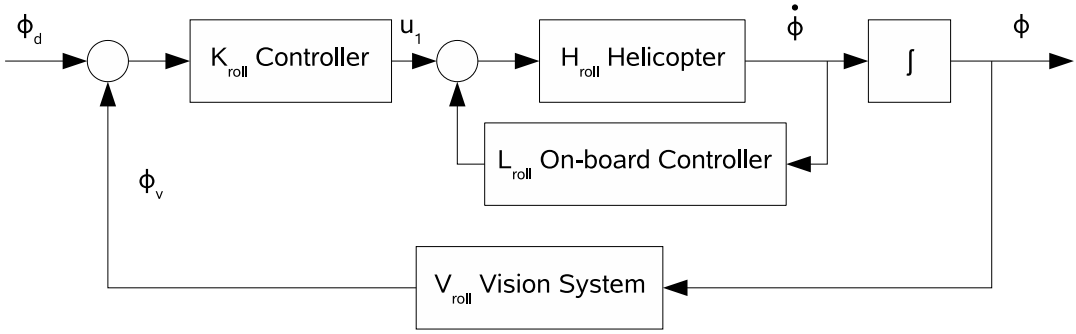


Figure 7.4: The control loop for MAV roll (ϕ).

The frequency response of the open loop system from u_1 to ϕ_v was measured and is shown in Figure 7.5. To achieve a higher bandwidth of control, more feedback gain is required and the open loop resonance clearly visible will cause destabilisation at high gains. To remove the effect of this resonance, a notch filter is employed:

$$N(z) = c\,\frac{1 - 2\lambda\cos(f)z^{-1} + \lambda^2 z^{-2}}{1 - 2\mu\cos(f)z^{-1} + \mu^2 z^{-2}}\ , \tag{7.2}$$

(expressed in the z-domain) where $\lambda = 0.95$, $\mu = 0.8$ and c is a normalising constant:

$$c = \frac{1 - 2\lambda\cos(f) + \lambda^2}{1 - 2\mu\cos(f) + \mu^2}\ . \tag{7.3}$$

The frequency f was set at 0.44 which corresponds to 3.5Hz, and the effect of the filter can be seen in Figure 7.6.

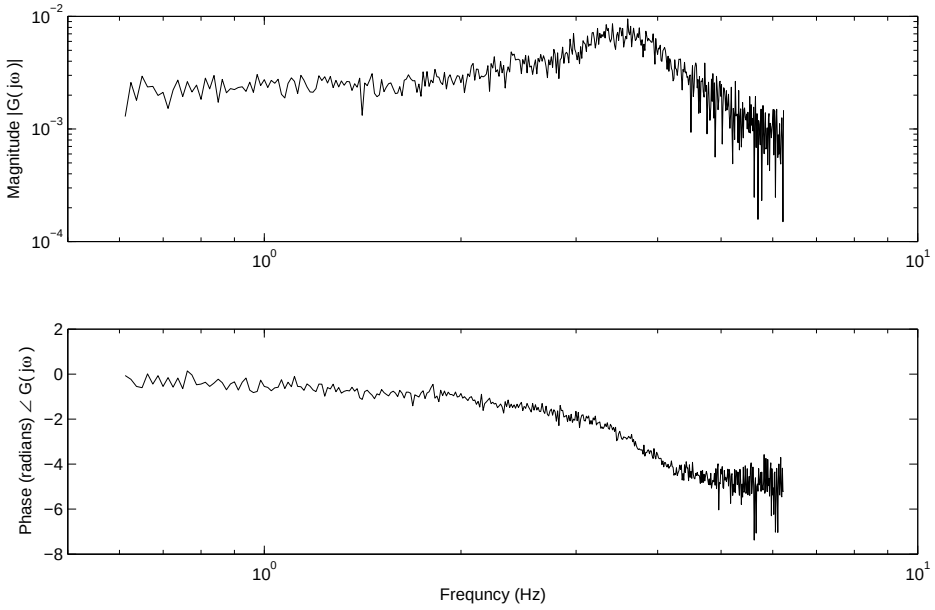


Figure 7.5: The frequency response of the open loop system from u_1 to ϕ_v.

With this flattened response, a high proportional feedback gain can be used and the final controller is:

$$K_{roll} = 4000\left[c\,\frac{1 - 2\lambda\cos(f)z^{-1} + \lambda^2 z^{-2}}{1 - 2\mu\cos(f)z^{-1} + \mu^2 z^{-2}}\right]\ , \tag{7.4}$$

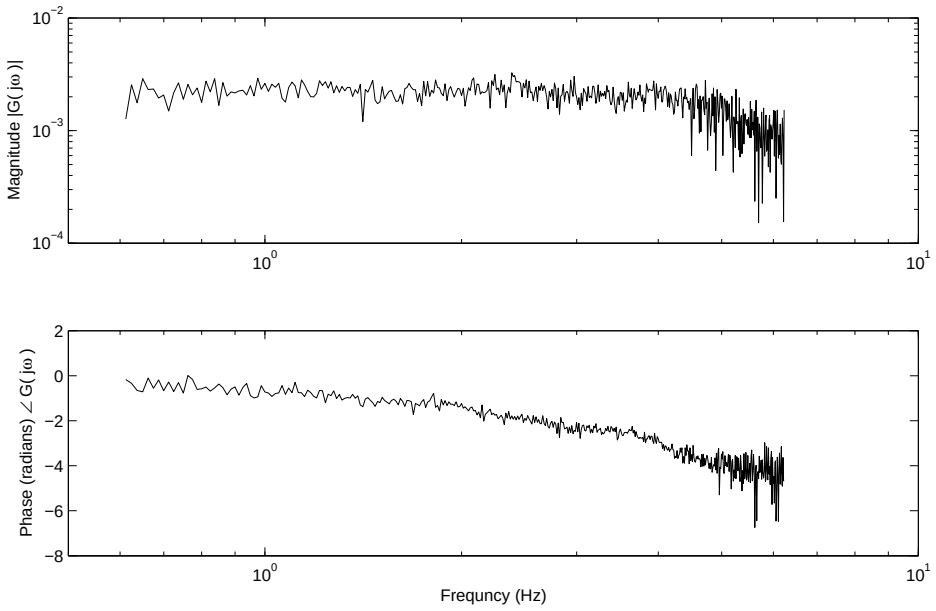


Figure 7.6: The frequency response of Figure 7.5 flattened by the notch filter given in Equation 7.2.

with c, λ, μ and f as Equation 7.3.

Using the flexibility of the interface described in Appendix B, it was possible to perform test flights where the overall power (u_3) was manually controlled, yaw was automatically controlled and the position of the roll and pitch joysticks (which normally control the first differential of these quantities) gave the roll and pitch demands for the controller just described. Such test flights were used to demonstrate the closed loop stability of the controller under resonant demands and external disturbances.

7.2.4 Position Control

The lateral (x, y) position of the MAV is controlled by adjusting its roll and pitch. The combined effect of the MAV's rotors will always be to produce a lift acceleration approximately equal to g, in a direction normal to the plane of the

MAV. A small angle of roll ϕ will therefore result in a lateral acceleration of $\ddot{x} = \phi g$. A nested controller [Buskey *et al.* 2003] is used to adjust the MAV's attitude in order to correctly position the helicopter, as shown in Figure 7.7.

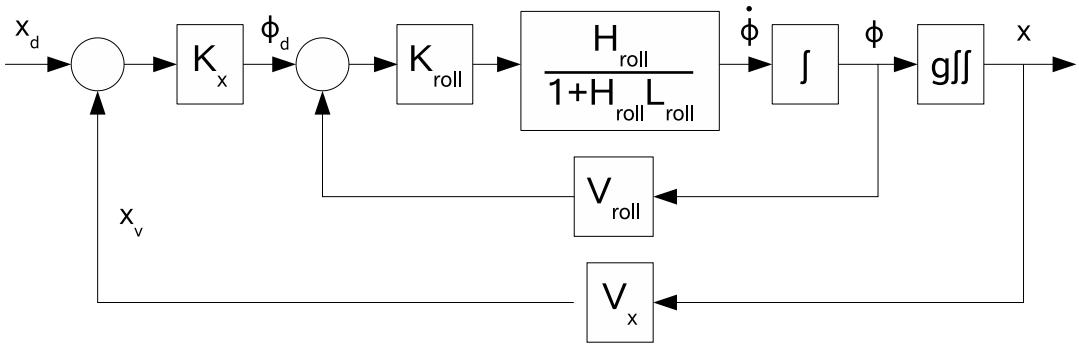


Figure 7.7: The nested control loop for MAV lateral position (x).

The controller chosen uses derivative terms to dampen the resonance of the second order system and it is these terms that require the accurate estimates of velocity discussed in Chapter 6. Additionally it is necessary to include some integral effects to remove any steady-state position error caused by roll and pitch trim[1]. The controller used is therefore:

$$\phi_d = 0.5(x_d - x_v) + 0.3(\dot{x}_d - \dot{x}_v) + 0.005 \int (x_d - x_v)dt \tag{7.5}$$

This controller is not expressed as a z-domain transfer function, since $\dot{x}_v$ is *not* obtained by a finite difference approximation but rather using the filtered estimates. The integral is however performed using a finite sum approximation.

7.2.5 Height Control

Controlling the height of the MAV provides the greatest challenge for three reasons. Firstly, it is hard to measure the system response, particularly at low frequencies, since the safe working volume is limited. Secondly, the ground effect

[1]In particular, roll and pitch trim errors are caused by errors in camera alignment affecting the system's definition of vertical.

means that the system response changes with height. Lastly, the response varies noticeably with the battery charge level.

When operating above the ground effect, the control output u_3 is approximately proportional to vertical acceleration. To remain at constant height, a significantly non-zero value of u_3 is therefore required to counteract gravity. This value must be provided by an integral term in the controller and this generates an additional difficulty when performing test flights. If the PC has complete control of u_3, the position of the corresponding joystick on the radio control transmitter is arbitrary. If the test pilot needs to take over control of the helicopter, upon switching to manual control the helicopter is likely to fall to the ground because the joystick is in the 'off' position. To prevent this it is necessary to ensure that the joystick is kept reasonably close to the current value of u_3. This is achieved by limiting the value of the integral term in the controller to a threshold above or below the current joystick position. A slider bar on the PC display shows the pilot the difference between the integral term and the current joystick location so that the pilot can modify the joystick position to keep the difference small. Providing the difference never reaches the threshold, the pilot will have no effect on the automatic control, yet is in a good position to take over control if necessary.

Typical flights however occur in or close to the ground effect. At very small heights, the control output u_3 is approximately proportional to vertical height, since the ground provides a negative feedback effect. This feedback effect is insufficient to stabilise heights above about 0.3m, but clearly continues to contribute to the open-loop for higher positions. Attempts to accurately determine the open-loop response failed due to these significantly non-linear effects and hence no forward prediction model is used for height (see Section 7.2.1). However, manual parameter adjustment has led to the controller:

$$u_3 = 100(z_d - z_v) + 300(\dot{z}_d - \dot{z}_v) + 5\int (z_d - z_v)dt \tag{7.6}$$

which provides a stable and reasonably fast response in most conditions.

7.3 Results

Figures 7.8 and 7.9 show plots of the lateral position (and height) recorded during trial flights with a step change in the demand position. Ground truth measurements were obtained using a remote video camera monitoring infra-red LEDs mounted on the MAV, as described in Section 6.4. These plots demonstrate both the accuracy of the visual tracking measurements when compared with ground truth and also the accuracy of the control loop in maintaining a demanded position.

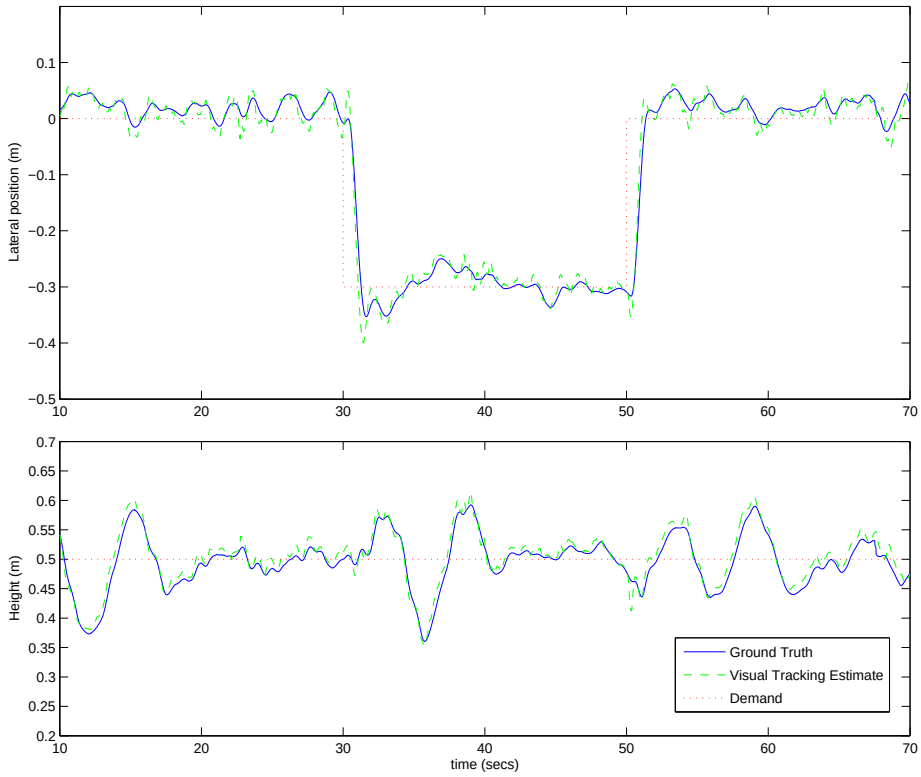


Figure 7.8: Results from a 0.3m step change in lateral position demand.

The 0.3m step change (Figure 7.8) is achieved with only a small overshoot and no significant oscillation. With a 0.5m demand change (Figure 7.9) the overshoot and

oscillations are more noticeable; these could be reduced by increasing the velocity feedback but at the expense of a slower response to more moderate disturbances. The control of height appears to be unaffected in both cases. The lateral overshoot visible for the 0.5m step change is interesting because it demonstrates the multi-modal filtering described in the previous chapter. At the peak of the overshoot, the visual tracking estimate significantly (by 12cm) over-estimates the lateral error. The is due to an initial strong belief in the constant velocity model. After a few further readings however, significant evidence is available to believe a large acceleration occurred, and the filtered estimate snaps back to be close to ground truth. This effect is more clearly visible in the close-up of Figure 7.10.

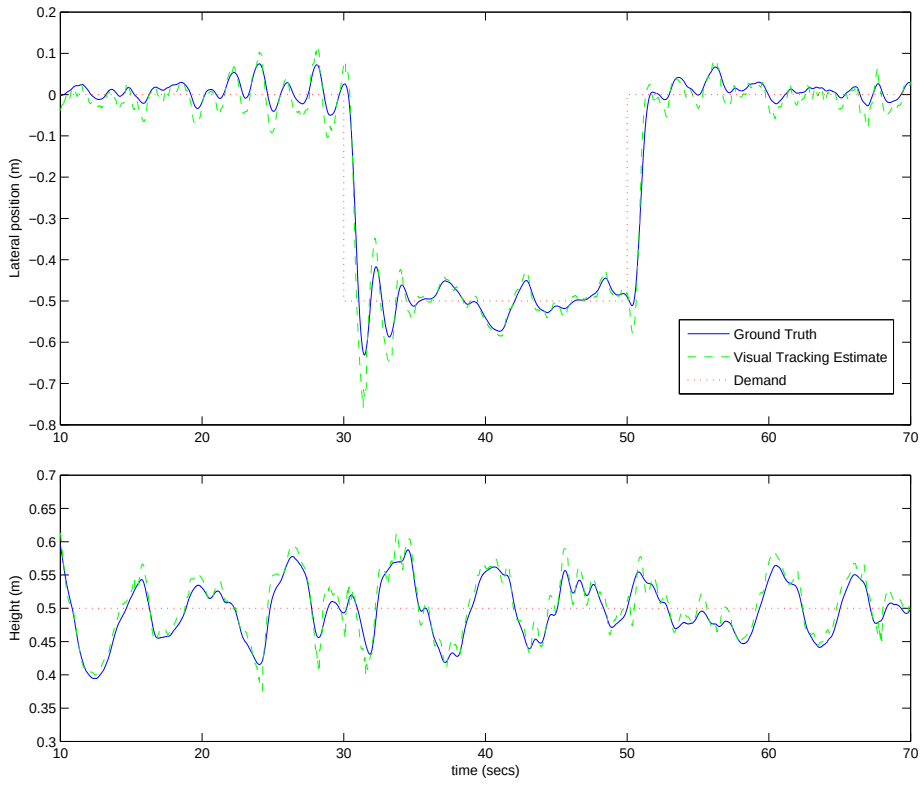


Figure 7.9: Results from a 0.5m step change in lateral position demand.

No results are presented for yaw control since ground truth measurements were

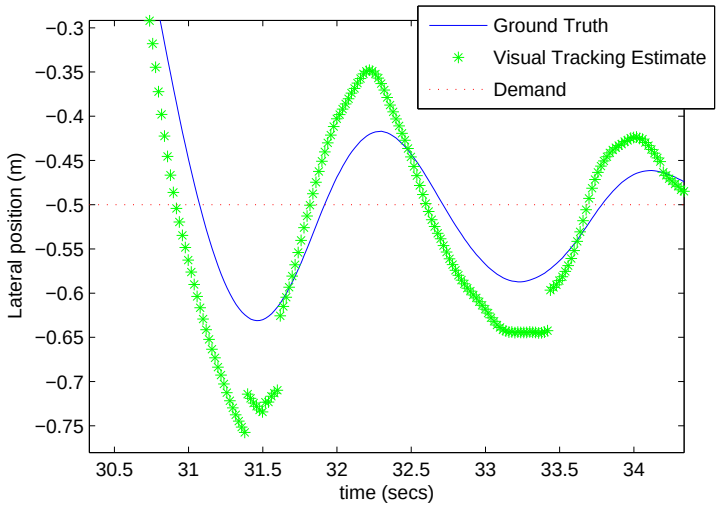


Figure 7.10: Close-up of Figure 7.9 showing the multi-modal filter initially overshooting the ground truth measurement but then jumping back to an accurate estimate once sufficient evidence is available.

difficult to obtain from the infra-red LED setup and observations confirm that yaw is controlled with ample accuracy and stability margins.

Figure 7.11 shows the response to a 0.2m step change in demanded height, performed across the boundary of the ground effect. A significant overshoot is clearly visible in the upwards direction, but not as the demand drops back down. This shows the non-linear effect of the ground and its effect in stabilising low level flight. No steady state error in the lateral position is introduced by the change in height, but the lateral dynamic response is noticeably more sedate at the increased height.

Figure 7.12 and Figure 7.13 show the helicopter flying in the lab and in a corridor environment. In both cases ramped position demands are successfully executed. Empirically, the accuracy of the automatic control exceeds that of a human pilot's attempts at manual control and in the vast majority of test flights no manual intervention is required. The controller is sufficiently stable that it can tolerate the helicopter being physically pushed away from the demand position during flights and attempts to manually excite resonance by physically pushing the MAV fail. Typically, problems occur due to failure of the tracking system rather than

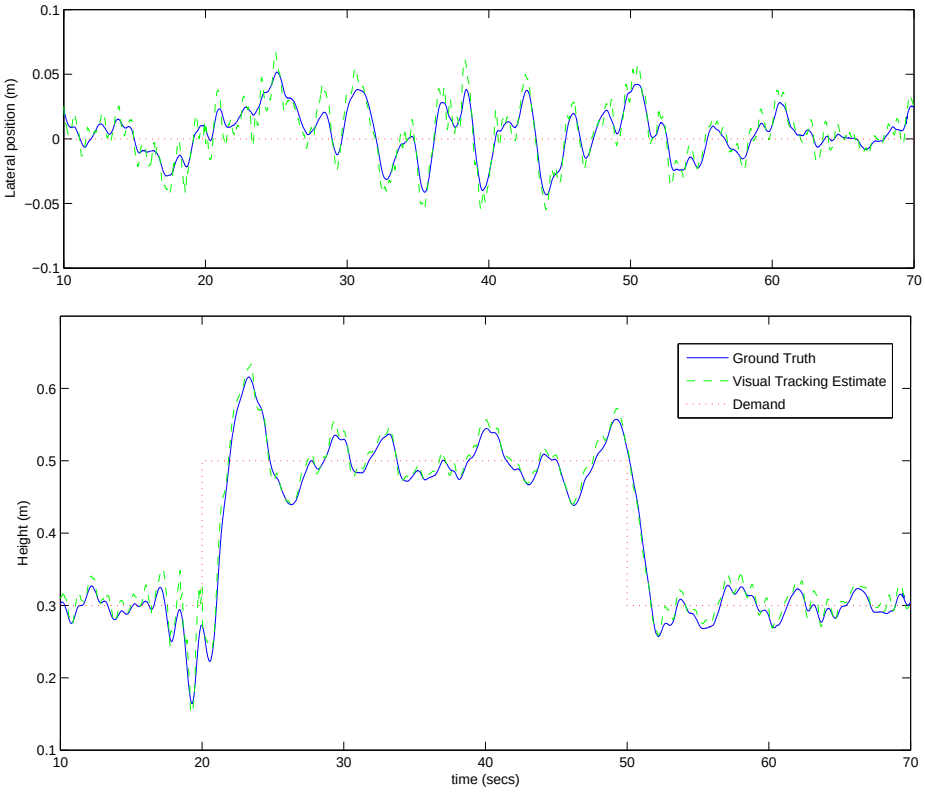


Figure 7.11: Results from a 0.2m step change in height demand.

the control loop, although for height demands below 0.3m (within the ground effect) flights currently become unstable as the battery power drops. Tracking is typically reliable in most areas of the test environment, but becomes prone to failure in certain areas where the view from the camera provides a weak constraint in a certain direction of camera motion. For pre-planned routes however, the success rate of flights is very high.

7.4 Conclusions

This chapter presented the real-time operation of the tracking and filtering systems described previously, together with a control loop to automatically fly the MAV. The tracking and filtering system is highly parallelisable and its real-time operation can be significantly aided through the use of a multi-threaded application running on a dual-processor machine. Mutual exclusion and synchronisation were found to cause difficulties when using multiple threads but many of these problems were avoided through the use of command objects and a multi-threaded queuing system.

The control of the four-rotor helicopter, given accurate pose and velocity estimates, is mostly straightforward and reasonable stability margins can be achieved. Height control remains a challenge when operating within the ground effect, and reliable stability in this regime requires further work. For most routes within the constraints of tracking however, the current system is able to repeatedly perform successful automatic flights with no manual intervention.

For most test flights, take-off and landing were effectively manually controlled due to the power limiting described in Section 7.2.5. However, with careful manipulation of the power control joystick it is possible to perform flights where the computer has unrestricted control for the entire flight, including take-off and landing. This confirms that take-off and landing present no significant challenges over those normally present when operating within the ground effect and shows that complete automatic flights are possible.

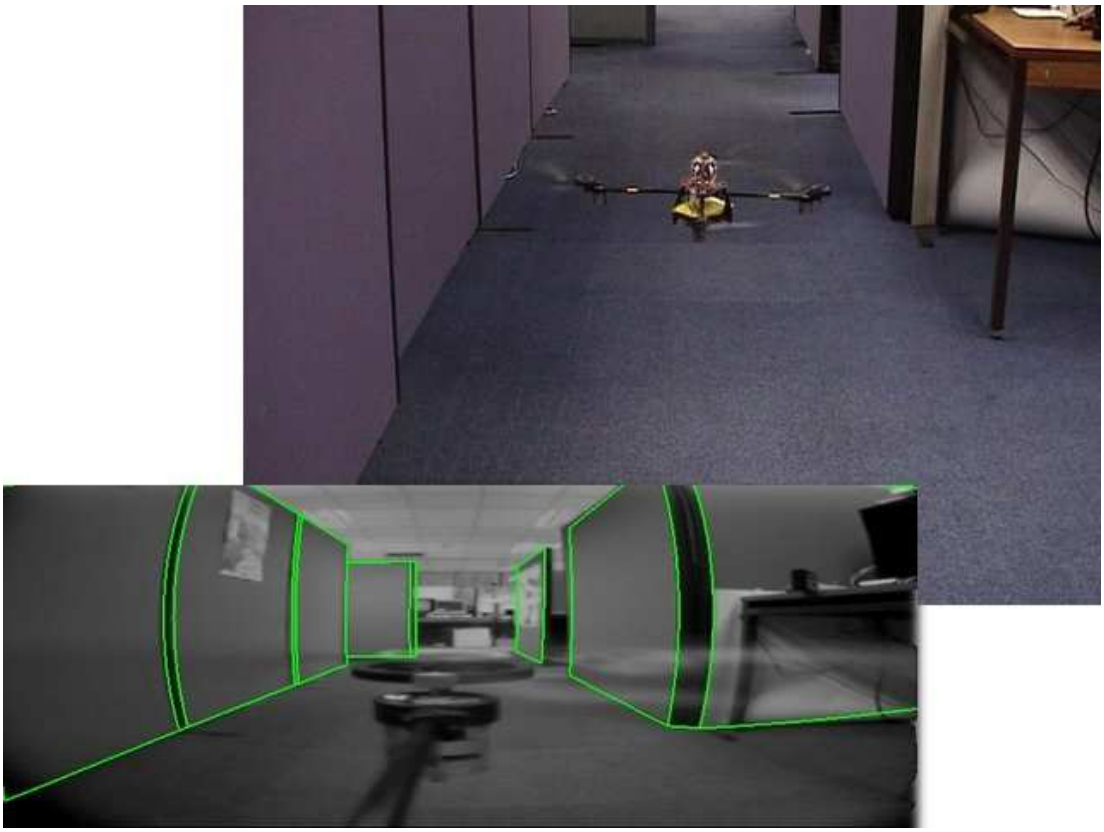


Figure 7.12: The helicopter automatically flying in a lab and an insert showing the tracked image from the onboard camera.

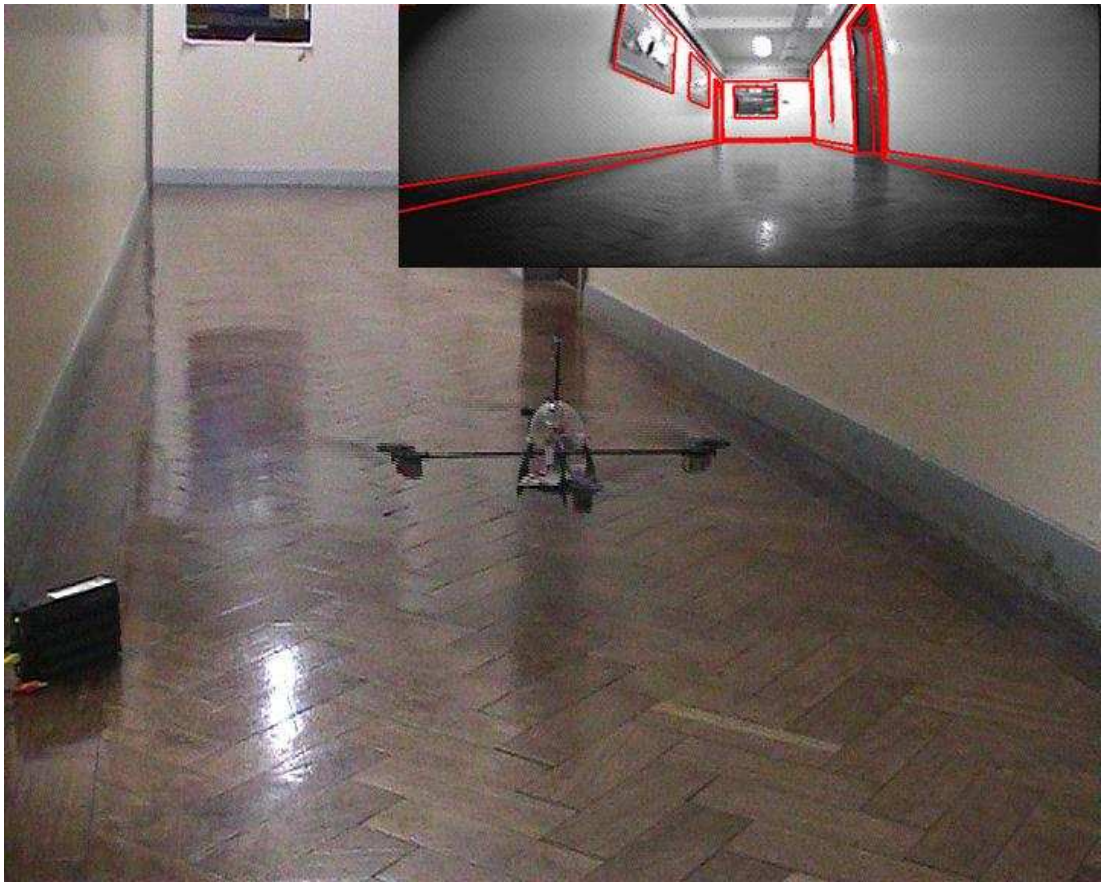


Figure 7.13: The helicopter automatically flying along a corridor and an insert showing the tracked image from the onboard camera.

Chapter 8

Conclusions

8.1 Summary

This thesis investigated the application of visual tracking to the automatic guidance of a model helicopter. The light weight and payload capacity of this indoor MAV led to several significant challenges. Employing just a sub-miniature, transmitting video camera allows the pose of the helicopter to be determined with only 9 grams of added equipment. However, the poor quality images obtained suggest the use of an edge-based tracking system with a 3D model of the MAV's surroundings, since edges are particularly stable features. Moreover, the light weight of the MAV means that external disturbances have a large, unpredictable, effect on its motion. This means that matching assumptions made by previous edge-based tracking methods are invalid and this often leads to tracking failure.

This tracking difficulty is addressed with novel techniques for generating multiple pose hypotheses and for doing so efficiently to ensure real-time operation. The resulting system is able to maintain track of the helicopter's pose, despite the poor image quality and significant motion disturbances, throughout the majority of trial flights.

Although the tracking system provides reliable estimates of the MAV's pose,

stable control of position also requires reasonable estimates of velocity and these cannot be reliably obtained directly from the video image. Filtering the noisy pose estimates in the presence of non-Gaussian motion disturbances, without introducing large time-lags, is challenging. However, a multi-modal filter is able to consider occasional large motion disturbances and, once significant evidence of a disturbance is available, can switch modes to obtain an up-to-date velocity estimate.

Finally, the fully automatic control of the MAV using just the information from the onboard video camera is presented. In most cases the system is able to repeatedly fly the helicopter along a route, with no manual intervention.

8.2 Contributions

This thesis has made the following contributions.

- A method to generate a sampled representation of a multi-modal distribution directly from the measurement data. The density of particles is higher near the peaks of the distribution and hence the method is computationally efficient. The system is particularly beneficial when applied in a hierarchical approach and it has been demonstrated to improve the robustness of an edge-based tracking system.

- An extension to a texture change-point detector which correctly allows it to detect multiple change-points along a single scanline. This allows the detector to be used as part of tracking system which admits multiple hypotheses for the location of edges in an image.

- A method to aid high-dimensional parameter estimation problems through the automatic generation of measurement clusters. The measurement clusters can be treated independently and each gives a parameter estimation task of reduced dimensionality. The combination of good results from the

clusters rapidly leads to the complete solution. The system has been demonstrated to significantly reduce the computational cost of multiple-hypothesis edge-based tracking.

- A multi-modal likelihood filter which obtains accurate yet responsive velocity results from noisy pose estimates. By allowing a more accurate acceleration model than is possible with a Kalman filter, the multi-modal filter is able to produce similarly accurate velocity estimates but can respond much more rapidly to occasional large accelerations.

- The demonstration of completely automatic MAV control. The contributions just described provide an accurate and reliable method for obtaining, in real-time, the pose and velocity of the MAV from a single, onboard video camera. A nested controller is able to use these estimates to automatically hover and fly the MAV.

8.3 Further Work

Although the automatic system works reliably in several situations, its development and testing revealed a number of areas which could benefit from further work.

- A system relying only on frame-to-frame tracking will never be robust to all disturbances and will always need initialising. A method to initialise the pose of the MAV on system startup would be necessary for it to be used by untrained operators. If such a method can also be used to recover from tracking failures with only a minimal time delay, in many cases this may be sufficient to avoid the helicopter crashing.

- The texture change-point detector presented in Chapter 4 provides a useful tolerance to textured scenes at only a modest computational cost. However, the underlying texture model does not consider the magnitude of any intensity change at a change-point and this causes difficulties with smoothly

ramping regions in the image. Section 4.5 describes one method for avoiding some of these difficulties but more rigorous inclusion of intensity magnitude information into the texture model would provide significant benefits.

- The system has been heavily tailored to indoor flights, where the MAV's surroundings are largely polyhedral, typical camera views provide good constraints on the full six dof pose and where the MAV's position must be controlled very accurately to avoid collision. Typical outdoor flights present a different set of challenges and further work would be required for reliable outdoor tracking and control.

- There are clearly situations where the automatic flight of the MAV becomes difficult due to the ground effect or due to tracking difficulties. It would be interesting to incorporate knowledge of these difficulties into the system so that they can be automatically avoided or rapidly traversed. For example, there are situations where the pose of the MAV can not be accurately determined (e.g. when looking at a frontal parallel scene) yet such situations could be automatically predicted from the 3D model. Doing so might enable the MAV to rapidly pass through areas of uncertainty to locations where its pose can be tracked more accurately. Alternatively its heading (yaw) could be automatically adjusted to maintain a good camera view. This would allow more sophisticated routes to be followed.

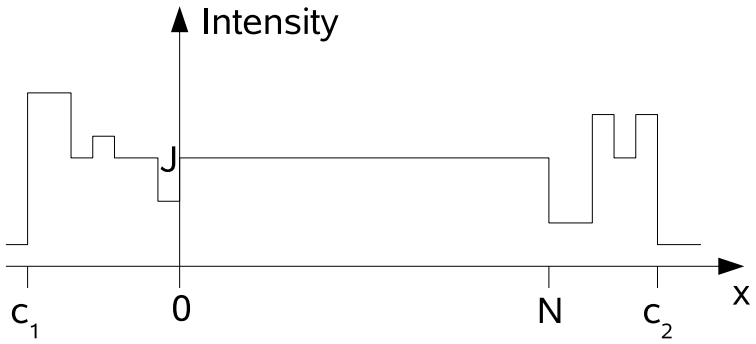
Appendix A

Proof of Theorem 1

Section 4.3 describes an algorithm for finding texture change-points in a 1D line scan. Theorem 1 stated '*Equation 4.15 will never be maximal if there is a change-point between pixels of the same binned intensity*'.

This appendix gives a proof that it is always more probable for a change-point to be at the end of a run of constant intensity than anywhere in the middle.

Consider a possible change-point lying somewhere between $x = 0$ and $x = N$:



- c_1: location of the previous change-point
- p_1: probability of the sequence up to the previous change-point

- c_2: location of the following change-point
- p_2: probability of the sequence after the following change-point
- I: the number of intensity bins
- $0 < x < N$: the constant intensity region
- J: intensity of the constant region
- n_1: occurances of intensity J in the region $c_1....0$
- n_2: occurances of intensity J in the region $N....c_2$

Equation 4.13 gives the probabilty of a sequence of length l, given a single texture, as:

$$\frac{(I-1)!\ \prod_{i=1}^{I}(o_i+1)!}{(I+l-1)!}\ . \tag{A.1}$$

So if:

$$\begin{aligned} k_1 &= \prod_{i=1..I, i\neq J}(o_{i1}+1)! \\ k_2 &= \prod_{i=1..I, i\neq J}(o_{i2}+1)! \end{aligned} \tag{A.2}$$

are the relevant factors for the pixels of intensity $\neq J$, the probability of the entire sequence, given a change-point at $0 \leq x \leq N$ is therefore:

$$P(S|x) = p_1\ \frac{(I-1)!\ k_1\ (n_1+x+1)!}{(I+x-c_1-1)!}\ \frac{(I-1)!\ k_2\ (n_2+N-x+1)!}{(I+c_2-x-1)!}\ p_2\ . \tag{A.3}$$

Grouping all the factors which don't depend on x into K and writing in terms of the negative log likelihood gives:

$$\begin{aligned} -\log P(S|x) &= K + \sum_{i=1}^{I+x-c_1-1}\log i - \sum_{i=1}^{n_1+x+1}\log i + \sum_{i=1}^{I+c_2-x-1}\log i - \sum_{i=1}^{n_2+N-x+1}\log i \\ &= K + \sum_{i=n_1+x+2}^{I+x-c_1-1}\log i + \sum_{i=n_2+N-x+2}^{I+C_2-x-1}\log i \\ &= K + \sum_{i=n_1+2}^{I-c_1-1}\log(i+x) + \sum_{i=n_2+N+2}^{I+C_2-1}\log(i-x)\ . \end{aligned}$$

Treating x as a continuous variable and differentiating the negative log likelihood twice gives:

$$\begin{aligned}
\frac{d}{dx} &= \sum_{i=n_1+2}^{I-c_1-1} \frac{1}{i+x} + \sum_{i=n_2+N+2}^{I+C_2-1} \frac{-1}{i-x} \\
\frac{d^2}{dx^2} &= \sum_{i=n_1+2}^{I-c_1-1} \frac{-1}{(i+x)^2} + \sum_{i=n_2+N+2}^{I+C_2-1} \frac{-1}{(i-x)^2} \quad < \quad 0 \ .
\end{aligned}$$

Since the second differential is always less than zero, there are no minima between 0 and N and the minimum of the negative log likelihood over the region $0 \leq x \leq N$ must always be at either $x = 0$ or $x = N$.

Appendix B

USBRC: Interfacing a PC to a RC transmitter

Section 1.2 gives an overview of the complete MAV control loop. Once the desktop PC has determined the helicopter motor speeds required, these are returned to the helicopter via a standard 4-channel radio control transmitter. The transmitter is equipped with a 'buddy-box' connection which provides both an output of the current joystick positions and an input which can be sent to the helicopter. A lever on the transmitter determines whether the helicopter is controlled by the external input or the standard joysticks.

An interface is required to link the buddy-box connection to the PC. Although commercial devices are available, these all operate to simply return the joystick positions to the PC or to allow the PC to control all the channels. For the development of the closed loop system two additional features were required. Firstly it is important that channels can be selectively controlled by the PC, with the remaining channels being manually controlled. In this way the controller for, say, yaw can be developed in isolation. Secondly, even when the PC is controlling all four channels, it is important that the PC can still monitor the joystick positions. If the pilot needs to take over from the PC it is vital that the joystick controlling overall power is approximately positioned. This can be

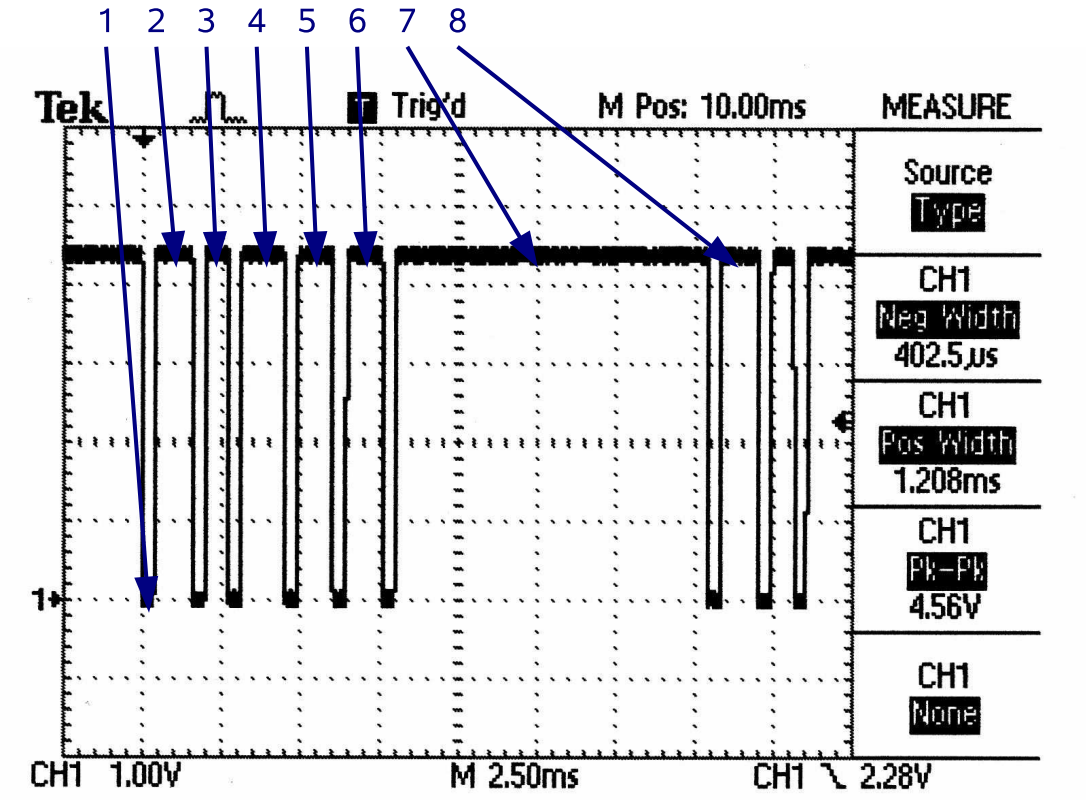
ensured by limiting the PC demanded power to within a tolerance of the manual joystick position.

B.1 Electronics

The electronic circuit must measure and generate the waveforms needed by the buddy-box connection on radio control transmitters. The majority of transmitters use 'pulse coded modulation' (PCM) where each demand is coded as the length of a positive pulse. Figure B.1 shows the waveform of this signal. Although it might have been possible to measure and generate this waveform using a PC parallel port, this would have required complicated real-time kernel level programming. A more flexible solution is to use a microcontroller which has dedicated timers to simplify the measurement and generation of such waveforms. This also ensures that valid signals are sent to the helicopter even if the PC crashes. An Atmel[1] microcontroller was chosen for its ease of development and satisfactory on-chip hardware (e.g. flash memory, timers, in-circuit debugging). Although the simplest solution would have been to link the microcontroller with the PC using RS232, chips such as the USBN9603 from National Semiconductor now make USB communications relatively straightforward and this allows use with portable computers which often lack a RS232 port.

The complete circuit schematic is shown in Figure B.3. Other than the microcontroller and USB interface, inverters are used to buffer the signals to and from the radio control transmitter and the signals are then routed to the relevant timer input and output pins on the microcontroller. The entire circuit is adequately powered by the 5v supply provided by the USB port. A PCB was designed (see Figure B.2) and assembled using surface mount components.

[1] ATMega32 - see `www.atmel.com`



1. Fixed length of 0.4ms
2. Varying length pulse for Channel 1
3. Varying length pulse for Channel 2
4. Varying length pulse for Channel 3
5. Varying length pulse for Channel 4
6. Channel 5 (unused)
7. Long synchronisation pulse $>$ 4ms
8. Next Channel 1 pulse

The length of the 5v section of each channel pulses determines its setting; lengths vary between 0.9ms and 1.6ms.

Figure B.1: Standard PCM Waveform.

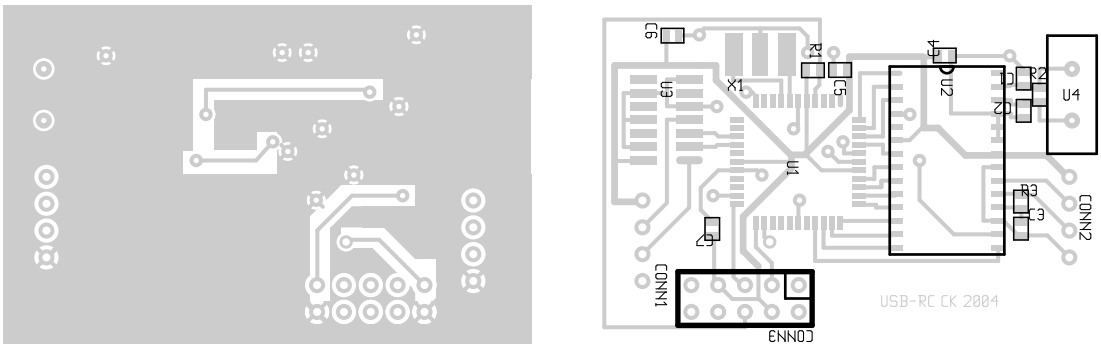


Figure B.2: USBRC PCB Layout.

B.2 Software

Two separate pieces of software are required; software running on the microcontroller and a Linux kernel module. The microcontroller software is written in assembly and is interrupt driven. A change of state on the input capture pin triggers an interrupt which reads the pulse length (measured by a hardware timer). The long synchronisation pulse shown in Figure B.1 is used to reset a channel counter. The required output is similarly generated by seeding a hardware timer with appropriate values and toggling the output pin when the timer overflows. The USBN9603 also generates a (lower priority) interrupt when it has new data for the microcontroller or when the microcontroller is free to send data to the PC (USB is entirely host controlled). Most of the microcontroller code is devoted to decoding and responding to the standard USB requests [Compaq et al., chap. 9] which are generated by the Linux kernel.

The kernel module employs the standard Linux USB subsystem and registers a character device (major number 241) which exposes four ioctls to turn on or off control of individual channels and to get or set a data structure containing the four channel settings.

The USB connection employs a control endpoint and two ‘interrupt’ endpoints (one for each direction) which provide on demand transfer of data packets. Error correction is provided by the USB layer, so a simple communication protocol can

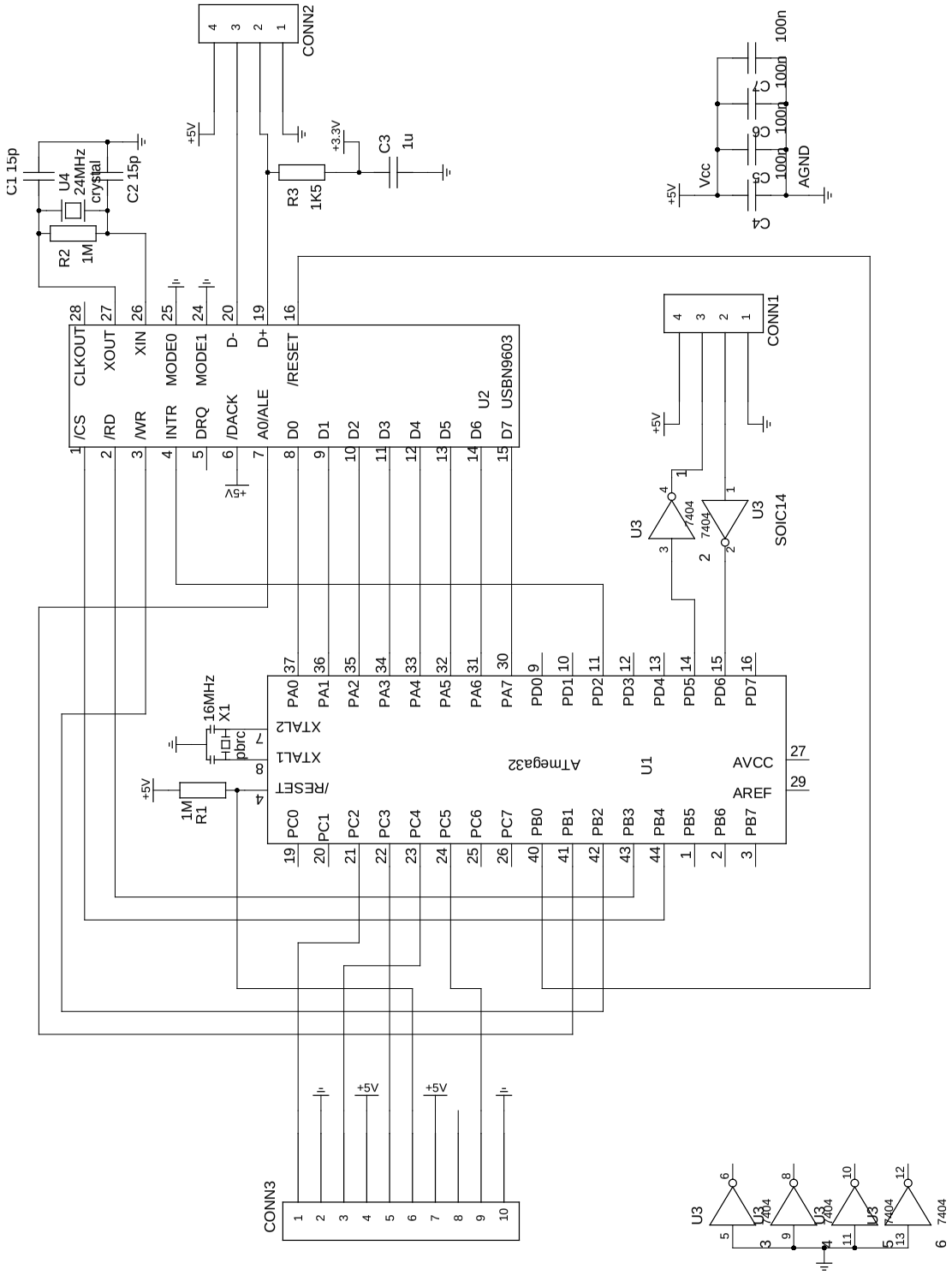


Figure B.3: USBRC Schematic.

be used. The control endpoint uses single byte packets to indicate which of the four channels should be controlled by the PC and which should be controlled by the transmitter joysticks. The interrupt endpoints use eight byte packets to transfer four words describing the four channel settings. The USB manufacturer id used is 0xffff and the product id is 0x01.

Author Index

AUTHOR INDEX

Bibliography

E. Altuğ (2003). *Vision Based Control of Unmanned Aerial Vehicles with Application to an Autonomous Four Rotor Helicopter*. Ph.D. thesis, University of Pennsylvania. 1.2, 2.8

O. Amidi (1996). *An Autonomous Vision-Guided Helicopter*. Ph.D. thesis, Department of Electrical and Computing Engineering, Carnegie Mellon University, Pittsburgh, PA15213. 1, 2.8

O. Amidi, T. Kanade & J. Miller (1998). Vision-Based Autonomous Helicopter Research at Carnegie Mellon Robotics Institute 1991-1997. In *American Helicopter Society International Conference*, Japan, paper no. T7-3. 1

N. Andreff, B. Espiau & R. Horaud (2002). Visual Servoing from Lines. *Intl. Journal of Robotics Research*, **21(8)**, 679–699. 2.8

M. Armstrong & A. Zisserman (1995). Robust Object Tracking. In *Proc. Asian Conference on Computer Vision*, vol. I, 58–61. 2.2.2, 2.6, 4.1

K. Arras, J. Castellanos & R. Siegwart (2002). Feature-Based Multi-Hypothesis Localization and Tracking for Mobile Robots Using Geometric Constraints. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 1371–1377. 2.3

S. Arulampalam, S. Maskell, N. Gordon & T. Clapp (2002). A Tutorial on Particle Filters for On-line Non-linear/Non-Gaussian Bayesian Tracking. *IEEE Transactions on Signal Processing*, **50(2)**, 174–188. 2.6, 2.7

BIBLIOGRAPHY

M. Berger, A. Thomas, G. Bachler, S. Scherer & A. Pinz (2000). 3D Model Based Pose Determination in Real-Time: Strategies, Convergence, Accuracy. In *Proc. 15th Intl. Conference on Pattern Recognition*, vol. 4, 567–570. 2.1

E. Bjorgo (1999). Digital Imagery in Global Disaster Information. *Bulletin of the American Society for Information Science*, **26(1)**. 1

G. Buskey, J. Roberts, P. Corke & G. Wyeth (2003). Helicopter Automation using a Low-Cost Sensing System. In J. Roberts & G. Wyeth, eds., *Proc. Australasian Conference on Robotics and Automation*. 2.8, 7.2.4

J. Canny (1986). A Computational Approach to Edge Detection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **8(6)**, 679–698. 2.5

T. Cham & R. Cipolla (1998). A Statistical Framework for Long-Range Feature Matching in Uncalibrated Image Mosaicing. In *Proc. Intl. Conference on Computer Vision and Pattern Recognition*, 442–447. 2.4

T. Cham & J. Rehg (1999). A Multiple Hypothesis Approach to Figure Tracking. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 239–245. 2.3, 2.6, 2.7, 6.3

M. Chen & M. Huzmezan (2003). A Simulation Model and $\mathcal{H}_\infty$ Loop Shaping Control of a Quad Rotor Unmanned Air Vehicle. In *Proc. IASTED Intl. Conference on Modelling, Simulation and Optimization*, 320–325. 1.2, 2.8

O. Chum & J. Matas (2002). Randomized RANSAC with $T_{d,d}$ Test. In *Proc. British Machine Vision Conference*, 448–457. 2.4

R. Cipolla & A. Blake (1992). Surface Shape from the Deformation of Apparent Contours. *Intl. Journal of Computer Vision*, **9(2)**, 83–112. 2.2.1

D. Comaniciu, V. Ramesh & P. Meer (2000). Real-Time Tracking of Non-Rigid Objects using Mean Shift. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 142–151. 2.2

Compaq et al. (2000). *Universal Serial Bus Specification*. Compaq Computer Corporation, Hewlett-Packard Company, Intel Corporation, Lucent Technologies Inc, Microsoft Corporation, NEC Corporation and Koninklijke Philips Electronics N.V., 2nd edn., `www.usb.org` last accessed Dec. 2005. B.2

I. Cox & S. Hingorani (1996). An Efficient Implementation of Reid's Multiple Hypothesis Tracking Algorithm and its Evaluation for the Purpose of Visual Tracking. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **18(2)**, 138–150. 2.3

A. Davison (2003). Real-Time Simultaneous Localisation and Mapping with a Single Camera. In *Proc. 9th Intl. Conference on Computer Vision*, 1403–1401. 2.2

J. Deutscher, B. North, B. Bascle & A. Blake (1999). Tracking Through Singularities and Discontinuities by Random Sampling. In *Proc. Intl. Conference on Computer Vision*, 1144–1149. 2.3

J. Deutscher, A. Blake & I. Reid (2000). Articulated Body Motion Capture by Annealed Particle Filtering. In *Proc. Intl. Conference on Computer Vision and Pattern Recognition*, vol. 2, 126–133. 2.3

J. Deutscher, A. Davison & I. Reid (2001). Automatic Partitioning of High Dimensional Search Spaces Associated with Articulated Body Motion Capture. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 669–676. 2.3, 2.7, 5.2.3

J. Dittrich (2002). *Design and Integration of an Unmanned Aerial Vehicle Navigation System*. Master's thesis, Georgia Institute of Technology, Atlanta, GA 30332. 2.8

T. Drummond & R. Cipolla (1999). Visual Tracking and Control using Lie Algebras. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2652–2659. 2.4, 3, 3.1.3, 3.2.4, 3.2.4, 4.4.1, 4.4.2, 5.2.3

BIBLIOGRAPHY

S. Ettinger, M. Nechyba, P. Ifju & M. Waszak (2002). Vision-Guided Flight Stability and Control for Micro Air Vehicles. In *Proc. IEEE Intl. Conference on Intelligent Robots and Systems*, vol. 3, 2134–2140. 2.8

M. Fischler & R. Bolles (1981). Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communcations of the A.C.M.*, **24(6)**, 381–395. 2.4, 4.2

D. Gennery (1992). Visual Tracking of Known Three-Dimensional Objects. *Intl. Journal of Computer Vision*, **7(3)**, 243–270. 2.2.1

I. Gordon & D. Lowe (2004). Scene Modelling, Recognition and Tracking with Invariant Image Features. In *Proc. Intl. Symposium on Mixed and Augmented Reality*, 110–119. 2.2

N. Gordon, D. Salmond & A. Smith (1993). Novel Approach to Nonlinear/Non-Gaussian Bayesian State Estimation. In *IEE-Proceedings-F*, vol. 140, 107–113. 2.3

C. Harris & C. Stennett (1990). rapid - A Video Rate Object Tracker. In *Proc. British Machine Vision Conference*, 73–77. 2.2.1, 4.3

S. Haykin, ed. (2001). *Kalman Filtering and Neural Networks*, 221–280. Wiley Publishing. 2.6

M. Isard & A. Blake (1998). condensation - Conditional Density Propagation for Visual Tracking. *Intl. Journal of Computer Vision*, **29(1)**, 5–28. 2.3, 2.7, 4.2.3, 4.3

F. Jurie & M. Dhome (2002). Hyperplane Approximation for Template Matching. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **24(7)**, 996–1000. 2.2

C. Kemp & T. Drummond (2004). Multi-Modal Tracking using Texture Changes. In *Proc. British Machine Vision Conference*, vol. 2, 859–868, Kingston. 1.3

C. Kemp & T. Drummond (2005). Dynamic Measurement Clustering to Aid Real-Time Tracking. In *Proc. 10th IEEE Intl. Conference on Computer Vision*, vol. 2, 1500–1507, Beijing. 1.3

S. Kirpatrick, C. Gelatt & M. Vecchi (1983). Optimization by Simulated Annealing. *Science*, **220**, 671–680. 2.4

N. Kita, A. Davison, Y. Kuniyoshi, I. Hara, T. Matsui, S. Rougeaux, T. Hori & S. Hirai (1999). Mobile Sensing Robots for Nuclear Power Plant Inspection. *Advanced Robotics*, **13(3)**, 355–356. 2.1

G. Klein & T. Drummond (2002). Tightly Integrated Sensor Fusion for Robust Visual Tracking. In *Proc. British Machine Vision Conference*. 2.2.2

G. Klein & T. Drummond (2005). A Single-Frame Visual Gyroscope. In *Proc. British Machine Vision Conference*, vol. 2, 529–538, Oxford. 6

M. La Civita, G. Papageorgiou, W. Messner & T. Kanade (2002). Design and Flight Testing of a High-Bandwidth $\mathcal{H}_\infty$ Loop Shaping Controller for a Robotic Helicopter. In *Proc. of the AIAA Guidance, Navigation and Control Conference*, Monterey, CA, AIAA-2002-4836. 1.2

D. Lowe (1992). Robust Model-Based Motion Tracking through the Integration of Search and Estimation. *Intl. Journal of Computer Vision*, **8(2)**, 113–122. 2.2.1

D. Lowe (1999). Object Recognition from Local Scale-Invariant Features. In *Proc. Intl. Conference on Computer Vision*, vol. 2, 1150–1157. 2.2.3

C.-P. Lu, G. Hager & E. Mjolsness (2000). Fast and Globally Convergent Pose Estimation from Video Images. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **22(6)**, 610–622. 2.2.3

J. MacCormick & A. Blake (1999). A Probabilistic Exclusion Principle for Tracking Multiple Objects. In *Proc. Intl. Conference on Computer Vision*, 572–578. 2.3, 2.7

BIBLIOGRAPHY

J. MacCormick & M. Isard (2000). Partitioned Sampling, Articulated Objects, and Interface-Quality Hand Tracking. In *Proc. European Conference on Computer Vision*, vol. 2, 3–19. 2.3, 5, 5.2.3

E. Marchand, P. Bouthemy, F. Chaumette & V. Moreau (1999). Robust Real-Time Visual Tracking using a 2D-3D Model-Based Approach. In *Proc. Intl. Conference on Computer Vision*, vol. 1, 262–268. 2.2.2, 2.4, 5

K. Mikolajczyk & C. Schmid (2005). A Performance Evaluation of Local Descriptors. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **27(10)**, 1615–1630. 2.2.3

T. Phong & R. Horaud (1995). Object Pose from 2-D to 3-D point and line correspondences. *Intl. Journal of Computer Vision*, **15(3)**, 225–243. 2.2.3

R. Prouty (1990). *Helicopter Performance, Stability and Control*. Robert E. Krieger Publishing Co., Malabar, Florida. 2.8

C. Rasmussen (2004). Texture-Based Vanishing Point Voting for Road Shape Estimation. In *Proc. British Machine Vision Conference*, vol. 1, 47–56. 5

I. Rekleitis (1996). Optical Flow Recognition from the Power Spectrum of a Single Blurred Image. In *Proc. Intl. Conference in Image Processing*, 159–166, Lausanne, Switzerland. 6

M. Ringer & J. Lasenby (2002). Multiple Hypothesis Tracking for Automatic Optical Motion Capture. In *Proc. European Conference on Computer Vision*, vol. 1, 524–536. 2.3

P. Rosin (1999). Robust Pose Estimation. *IEEE Trans. on Systems, Man and Cybernetics*, **29(2)**, 297–303. 2.2.3, 2.4

E. Rosten & T. Drummond (2005). Fusing Points and Lines for High Performance Tracking. In *Proc. 10th IEEE Intl. Conference on Computer Vision*, vol. 2, 1508–1515, Beijing. 2.2, 4.1

Y. Rui & Y. Che (2001). Better Proposal Distributions: Object Tracking Using Unscented Particle Filter. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 786–793. 2.6

P. Santana & J. Barata (2005). Unmanned Helicopters Applied to Humanitarian Demining. In *Proc. 10th IEEE Intl. Conference on Emerging Technologies and Factory Automation*, vol. 1, 729–738. 1

S. Se, D. Lowe & J. Little (2001). Vision-based Mobile Robot Localization and Mapping using Scale-Invariant Features. In *Proc. IEEE Intl. Conference on Robotics and Automation*, vol. 2, 2051–2058. 2.2

A. Shahrokni, T. Drummond & P. Fua (2004). Texture Boundary Detection for Real-Time Tracking. In *Proc. European Conference on Computer Vision*, vol. 2, 566–577. 2.5, 4.3, 4.3, 4.3

O. Shakernia, Y. Ma, T. Koo & S. Sastry (1999). Landing an Unmanned Air Vehicle: Vision Based Motion Estimation and Nonlinear Control. *Asian Journal of Control*, **1(3)**, 128–145. 2.8

H. Shim, T. Koo, F. Hoffmann & S. Sastry (1998). A comprehensive study of control design for an autonomous helicopter. In *Proc. 37th IEEE Conference on Decision and Control*, 3653–3658. 2.8

G. Simon & M.-O. Berger (1998). A Two-Stage Robust Statistical Method for Temporal Registration from Features of Various Types. In *Proc. Intl. Conference on Computer Vision*, 261–266. 2.2.2, 2.4

G. Simon, A. Fitzgibbon & A. Zisserman (2000). Markerless Tracking using Planar Structures in the Scene. In *Proc. Intl. Symposium on Augmented Reality*, 120–128. 2.2

B. Sinopoli, M. Micheli, G. Donato & T. Koo (2001). Vision Based Navigation for an Unmanned Air Vehicle. In *Proc. IEEE Intl. Conference on Robotics and Automation*, 1757–1765. 2.8

BIBLIOGRAPHY

S. Smith (1997). Reviews of Optic Flow, Motion Segmentation, Edge Finding and Corner Finding. Tech. rep., Oxford Centre for Functional Magnetic Resonance Imaging of the Brain. 2.5

B. Stenger, A. Thayananthan, P. Torr & R. Cipolla (2003). Filtering using an Tree-Based Estimator. In *Proc. 9th Intl. Conference on Computer Vision*, vol. 2, 1063–1070. 2.3

J. Sullivan, A. Blake, M. Isard & J. MacCormick (1999). Object Localization by Bayesian Correlation. In *Proc. Intl. Conference on Computer Vision*, vol. 2, 1068–1075. 2.3

S. Todorovic & M. Nechyba (2004). Towards Intelligent Mission Profiles of Micro Air Vehicles: Multiscale Viterbi Classification. In *Proc. European Conference on Computer Vision*, 178–189. 2.8

S. Todorovic, M. Nechyba & P. Ifju (2003). Sky/Ground Modeling for Autonomous MAVs. In *Proc. IEEE Intl. Conference on Robotics and Automation*, vol. 1, 1422–1427, Taipei, Taiwan. 2.8

C. Tomlin (1997). Control of Systems on Lie Groups. Tech. Rep. UCB/ERL M97/50, EECS Department, University of California, Berkeley. 3.2.3

B. Tordoff & D. Murray (2002). Guided Sampling and Consensus for Motion Estimation. In *Proc. European Conference on Computer Vision*, vol. 1, 82–98. 2.4, 3.2.5

P. Torr & D. Murray (1997). The Development and Comparison of Robust Methods for Estimating the Fundamental Matrix. *Intl. Journal of Computer Vision*, **24(3)**, 271–300. 2.4

P. Torr & A. Zisserman (2000). MLESAC: A New Robust Estimator with Application to Estimating Image Geometry. *Computer Vision and Image Understanding*, **78(1)**, 138–156. 2.4, 3.2.5, 4.2, 4.2

K. Toyama & A. Blake (2001). Probabilistic Tracking in a Metric Space. In *Proc. Intl. Conference on Computer Vision*. 2.2

L. Vacchetti, V. Lepetit & P. Fua (2003). Fusing Online and Offline Information for Stable 3D Tracking in Real-Time. In *Proc. Intl. Conference on Computer Vision and Pattern Recognition*, vol. 2, 241–248. 2.2

L. Vacchetti, V. Lepetit & P. Fua (2004). Combining Edge and Texture Information for Real-Time Accurate 3D Camera Tracking. In *Proc. Intl. Symposium on Mixed and Augmented Reality*, 48–57. 2.2

R. van der Merwe, A. Doucet, N. de Freitas & E. Wan (2000). The Unscented Particle Filter. In *NIPS*, 584–590. 2.6

G. Welch & G. Bishop (1995). An Introduction to the Kalman Filter. Tech. rep., University of North Carolina, Chapel Hill, NC, USA. 2.6, 2.7

P. Wunsch & G. Hirzinger (1996). Registration of CAD-Models to Images by Iterative Inverse Perspective Matching. In *Proc. 13th Intl. Conference on Pattern Recognition*, 77–83. 2.2.3, 2.4

H. Zhang & J. Ostrowski (1999). Visual Servoing with Dynamics: Control of an Unmanned Blimp. In *Proc. IEEE Intl. Conference on Robotics and Automation*, vol. 1, 618–623. 1, 2.8

Z. Zhang (1997). Parameter Estimation Techniques: A Tutorial with Application to Conic Fitting. *Image and Vision Computing*, **25**, 59–76. 3.2.4

Z. Zhang (2000). A Flexible New Technique for Camera Calibration. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, **22(11)**, 1330–1334. 3.2.2