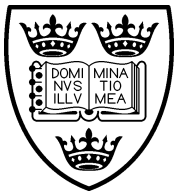
Probabilistic Methods for Enhanced Marine Situational Awareness



Charles Bibby

Worcester College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2009

Statement of Originality

This thesis is submitted to the Department of Engineering Science, University of Oxford, for the degree of Doctor of Philosophy. This thesis is entirely my own work, and, except where otherwise indicated, describes my own research.

Acknowledgements

Let me begin by thanking my supervisor Dr. Ian Reid for his help, inspiration and giving me the freedom to explore my ideas. It has been a pleasure to be a part of such a wonderful research group and I would like to thank everyone for their valuable advice, discussions and friendship.

I would like to acknowledge my sponsor company, Servowatch Systems and give special thanks to Steve Smith for his continual support, to Ian for teaching me how to program and deal with the black art of electronics hardware and to Stafford for his excitement and enthusiasm.

I would also like to thank my family. In particular, my dad and brother who are always there to help when it comes to testing equipment in the marine environment. My dad's craftsmanship and practical abilities have proved invaluable over the years.

Finally, a special thank you to Helena for putting up with me during the DPhil process. Helena has supported me in every way, not to mention her great proof reading skills.

Abstract

We present a system that uses *probabilistic methods* for enhanced situational awareness in marine environments. More specifically, we present significant contributions in the areas of Simultaneous Localisation and Mapping (SLAM) from the robotics community, and visual tracking from the computer vision community. We demonstrate how these theoretical contributions can be applied to the practical problem of marine surveillance.

Traditionally, SLAM has been used to produce static maps of the environment and dynamic objects have either been filtered out and ignored, or detected and tracked separately. In contrast, we show how dynamic objects can be included directly in the SLAM estimate. We propose a hybrid representation that combines point features, occupancy grids and cubic splines. The point features are used to represent small stationary objects, the occupancy grid is used to represent landmasses and cubic splines are used to represent the trajectories of dynamic objects.

We also present a new region-based, level-set framework, for visual tracking and segmentation. In contrast to all previous methods, we use the *pixel-wise posterior* when computing the foreground/background pixel membership, as opposed to the traditional pixel-wise likelihood. We show that this approach produces better behaved objective functions and hence provides more resilient visual tracking. We are able to track twelve or more objects in *real-time*, simultaneously estimating their position, rotation, scale, depth-ordering, figure-figure and figure-ground segmentations.

Finally, we describe a prototype system that combines our work on SLAM and visual tracking, enabling a mobile vehicle to be used for marine surveillance. This system produces a hybrid map of the environment that contains both metric and visual information. The metric information represents the position, speed, shape and size of objects. The visual information represents the appearance of objects and is acquired using a high-performance pan-tilt device, which we have designed, built and tested. We use the pan-tilt device to automatically make visual contact with all objects within sensor range. The prototype system has been demonstrated protecting a river in the Thames Estuary against potential security threats.

Preface

At the beginning of the 13th century mariners began plotting the first navigational charts (Portolan charts). These charts were used in conjunction with the magnetic compass to guide mariners safely from one port to the next, but relied on the mariners being able to see distinguishable landmarks. During the 15th century the mariner's quadrant (an early type of sextant) was invented, allowing the altitude of the north star (Polaris) to be measured and so it became possible to estimate latitude. Nearly three hundred years later in 1764, John Harrison succeeded where others had failed and devised a way of measuring longitude, the key component was his design for a clock that remained accurate even at sea (the rolling motion of a boat renders large pendulum based designs useless). By the end of the 18th century, your average mariner was able to estimate their position anywhere on the globe to within a few miles, using the sun, stars and a few man made instruments.

The second world war generated a need for detecting incoming hostile aircraft and ships. In response, Radio Detection and Ranging (RADAR) was developed. The principle of RADAR is to send out pulses of microwave radiation and to measure the time-of-flight for reflections to be received back, enabling accurate measurements of both the bearing and range of objects. There was also heavy development in the area of Sound for Navigation and Ranging (SONAR), which measures the distance between objects by timing sound waves in water or air. These technologies meant that mariners were now able to accurately measure relative distances and bearings between their own vessel and other objects.

Towards the end of the 20th century, the accurate measurement of time once again played a huge role in position finding. In 1993 the 24th Navstar satellite was launched and with four atomic clocks aboard, it completed the network of satellites that make up the Global Positioning System (GPS). This system can accurately measure a position anywhere on the globe to within a few metres and plays a huge role in modern life.

A common theme throughout maritime history is the desire to accurately estimate a vessel's location and the locations of objects in the surrounding area. Modern day

requirements are even more challenging and expect an information rich representation of the marine environment. Unmanned sensor networks are being deployed, generating the need for information engineering techniques to fuse incoming sensor data and automatically drive active sensors e.g. pan-tilt camera(s).

One such example is port security, where a number of sensors (RADARs, cameras, hydrophones and transponders) are distributed between various land and marine based platforms. The information from these sensors needs to be processed to form a single coherent picture of the entire port area, so that security operators can make decisions quickly without having to deal with huge volumes of raw sensor data.

Motivation

I am a keen sailor and on too many occasions have experienced first hand the pressure building as you find yourself slowly losing control of a situation at sea. It normally begins with tiredness and adverse weather conditions that make normal day-to-day book keeping tasks difficult to stay on top of. Then something unexpected gets thrown in: an unexpected or missing buoy or mark, an engine failure, a crew injury or an equipment failure. Your attention gets shifted to the unexpected event and your ability to process information relating to other events reduces. If you are unlucky, then another event may go undetected, for example: drifting off course, an accidental gybe[1], failing to recognise a collision threat or simply forgetting to look where you are going. This type of vicious cycle tends to worsen and often, an initially minor event, escalates into a serious one.

The motivation behind this dissertation is to provide mariners with a tool set that enhances their situational awareness and reduces their workload. The idea being that by reducing their workload during difficult or challenging situations, it will reduce the chance of a vicious cycle setting in and serious or potentially fatal accidents from occurring. A secondary benefit is that the reduced workload can also increase productivity for commercial, military or public services in the marine environment e.g. harbour patrol, search and rescue, naval or coastguard activities.

Although my personal experiences are in the context of leisure sailing, accident investigation reports and articles in the press show that similar experiences are often behind accidents and financial losses in the commercial, military and public sectors of the marine industry.

[1]This results in a violent movement of the boom that can fatally injure crew members.

Contents

Chapter 1

Introduction

1.1 Objective

The objective of this dissertation is to apply probabilistic methods to the marine industry, with the goal of enhancing situational awareness. Specifically, we use the idea of SLAM from robotics, and visual tracking from computer vision, to build an enhanced map of the nearby environment. This enhanced map includes both *metric* (positions/velocities/sizes) and *visual* (what it looks like) information, for stationary surface objects, dynamic surface objects and landmasses. This has the benefit of *tying visual and metric information together,* so that a marine operator can see at a glance both where objects are in relation to a vessel and what they look like. In general the vessel, which we will refer to as the vehicle, may also be dynamic. This makes the problem significantly harder as the egomotion (the vehicle's motion through the environment) must be estimated in conjunction with the hybrid map. Although this dissertation concentrates mainly on marine RADAR data and colour video, the principles developed could be applied to a much broader variety of sensor modalities.

Our objective can be written down mathematically with the following probabilistic statement:

$$P(\mathbf{X}, \mathbf{M}|\mathbf{Z}) = \frac{P(\mathbf{Z}|\mathbf{X}, \mathbf{M})P(\mathbf{X}, \mathbf{M})}{P(\mathbf{Z})}, \tag{1.1}$$

where: $\mathbf{X}$ is the vehicle's location and speed; $\mathbf{M}$ is the enhanced map i.e. the positions, speeds, shapes, sizes and visual appearance of other objects in the environment and

Z is the collection of sensor data i.e. marine RADAR and colour video. For readers not familiar with Bayes rule, Equation (1.1) will be explained in Chapter 2.

1.2 Challenges

Below is a list of the most significant challenges when estimating metric and visual information, using a marine sensor platform i.e. a boat:

Dynamic platform: The sensor platform is moving, which means that the sensors' egomotion needs to be estimated in conjunction with the metric information of the surrounding objects. This is why a SLAM solution is required, as opposed to if the sensors were stationary, in which case the objects could be mapped independently. This also means that any visual tracking using pan-tilt devices needs to be stabilised to compensate for the egomotion. This could be achieved with inertial sensing, visual sensing or as we do in this dissertation, a combination of the two.

Dynamic environment: Typically, SLAM systems filter out responses generated by dynamic objects in the environment so that the resultant map only contains stationary objects. In contrast, this dissertation addresses the problem of including dynamic objects directly within the SLAM estimation.

Intermittent measurements: Marine RADAR sensors often give intermittent responses from objects in the environment, especially in adverse weather. This means that the estimation has to be able to fuse intermittent responses over long periods of time.

Clutter: Marine RADAR also generates large amounts of clutter (false-positives); therefore the estimation techniques need to be able to handle clutter without detrimental consequences to the quality of the estimated environment.

Agile motion: The objects we wish to visually track can exhibit agile motion in the camera view both due to egomotion of the vehicle the sensors are mounted on, but also due to the motion of the objects on the surface of the sea. This means that the visual tracking technique used has to be able to handle agile motion.

Shape changes: In order to visually track an object for prolonged periods of time, the system will need to be able to handle changes in the object's projected shape in the image. This is because the objects can carry out manoeuvers i.e. out-of-plane rotations, which change their projected shape on the camera's image plane.

Appearance changes: The visual appearance of objects can change over time, both due to environmental lighting conditions and because a manoeuvering object causes different aspects to be exposed or hidden e.g. a boat may be blue down one side and red down the other.

Tracking drift: To combat the previous two challenges a solution is required that performs online learning about the objects being tracked. Online learning introduces the problem of tracking drift, which often results in tracking system failures.

Inter-object occlusion: If another object occludes the object being visually tracked, then often either complete failure will occur or the tracker may end up tracking the wrong object.

System integration: Considerable amounts of hardware and system integration was required for the objective of this dissertation, which is notorious for generating unforeseen problems.

Real-time: The solutions to the previous challenges have to be capable of processing the incoming data in real-time, in order to be useful in the real-world. Unless otherwise stated real-time is taken to be video rate, which we define as 30Hz or greater.

1.3 Contributions

Below is a list of the contributions found in this dissertation (in the order they appear):

SLAM in dynamic environments: Dynamic objects are incorporated in a SLAM framework using sliding window estimation, reversible decision making and generalised expectation maximisation. The temporal sliding window allows the system adequate time to get model-selection, data-association and clutter rejection correct before performing marginalisation. The results show that this allows a consistent estimate to be maintained, even in the presence of dynamic objects and significant amounts of clutter.

Hybrid representation: A hybrid representation is used to represent the marine environment, with point features representing small stationary objects, an occupancy grid representing landmasses and cubic splines representing the trajectories of dynamic objects. This hybrid representation is essential in providing a SLAM system that actually works in the real environment. All too often SLAM systems model certain aspects of the environment using point features, where some other representation

would offer superior performance.

Cubic splines are used to represent trajectories: Cubic splines are used to represent the trajectories of dynamic objects in the environment. This has three key benefits: (i) the number of states that need to be estimated can be reduced by representing a trajectory using spline sections; (ii) because the spline is continuous, it becomes natural to handle asynchronous measurements running at different frequencies rather than trying interpolate and/or extrapolate measurements to match a time-step present in the estimation framework and (iii) the continuous output makes it possible to re-render sensor data at a sub-scan resolution to compensate for the egomotion during data-acquisition.

Pixel-wise posterior tracking: A new visual tracking method based on pixel-wise posteriors, as opposed to pixel-wise likelihoods, is developed and quantitative results show that it provides better behaved objective functions for visual tracking. This tracking algorithm learns both the shape and appearance of an object online and the problem of tracking drift is tackled. This allows previously unseen objects to be tracked for long periods of time with minimal prior information.

Computing the depth-ordering of multiple objects: The pixel-wise posterior tracking algorithm is generalised to handle multiple interacting/occluding objects. A depth posterior is computed at each time-step, allowing the tracking algorithm to account for inter-object occlusions. The method is able to simultaneously estimate the position, scale, rotation, depth-ordering, figure-ground and figure-figure segmentation for up to twelve interacting objects in *real-time*.

System integration: We combine the theoretical contributions using good practical engineering, to produce a prototype system that can be used for situational awareness in marine environments. This system was demonstrated live over a period of two days to an audience of industry experts, governmental advisors, the navy, special forces, the police and politicians. The system was demonstrated protecting a river in the Thames Estuary against potential security threats.

Active vision system: Part of this real-world application was the design, build and test of a high performance pan-tilt device. This device is capable of panning 360 degrees and fixating to 1/1000th of a degree within 600ms. This type of performance is required to quickly saccade between objects in the environment, so visual information can be acquired efficiently.

Real-time performance: All of the methods and techniques presented in this dis-

sertation run in real-time i.e. 30Hz or greater on standard Personal Computer (PC) hardware.

1.4 Preview

Figure 1.1 shows a screen shot from the prototype system developed using the methods presented in this dissertation. The left-hand pane shows the estimated metric information and the right-hand pane shows the associated visual information. The metric information is estimated using the hybrid mapping technique presented in this dissertation. The overlay in the top-right of the left-hand pane shows the live feed from the pan-tilt camera designed, built and tested as part of this DPhil. The pan-tilt automatically makes visual contact with the surrounding objects and acquires visual information. Closed loop control of the pan-tilt device, using the visual tracking methods described in this dissertation, can be used to obtain stabilised video feeds of objects of interest. These stabilised video feeds can then be used for further visual processing, for instance a visual recognition module or for human classification.

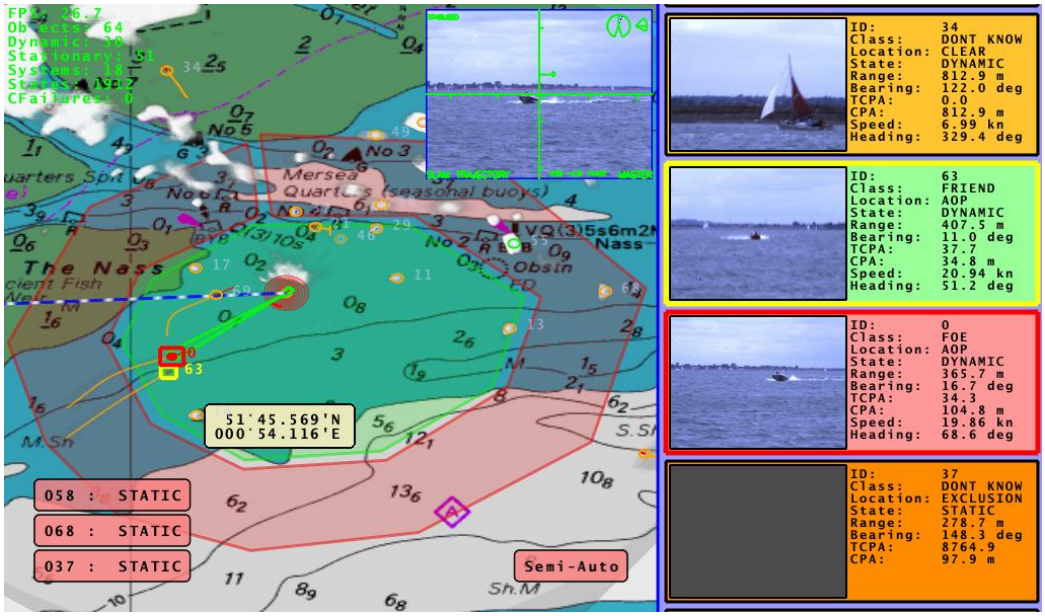


Figure 1.1: A screen shot taken from the prototype system built using the methods proposed in this dissertation. This system was demonstrated protecting a river in the Thames Estuary.

1.5 Outline

There are three main areas of contribution presented in this dissertation, which are indicated using three parts:

- **Part I:** describes the theoretical contributions for estimating the metric aspects of the environment.
- **Part II:** describes the theoretical contributions for visual tracking.
- **Part III:** describes how the theoretical contributions from Parts I and II are integrated into a real system.

The specific chapter-by-chapter break down is as follows:

- **Background Knowledge**
  - **Chapter 2:** introduces the notation, probabilistic rules and techniques used within this dissertation.
- **Part I: Estimating Metric Information**
  - **Chapter 3:** introduces the ideas behind SLAM and reviews the literature.
  - **Chapter 4:** describes our novel use of sliding window estimation and expectation maximisation to achieve reversible decision making. In particular, how to do reversible model-selection, reversible data-association and clutter rejection, allowing us to include dynamic objects within the SLAM framework.
  - **Chapter 5:** describes our hybrid representation for the marine environment, with point features representing small stationary objects, an occupancy grid representing landmasses and cubic splines representing the trajectories of dynamic objects.
- **Part II: Estimating Visual Information**
  - **Chapter 6:** describes our novel method for visual tracking that uses pixel-wise posteriors to represent the foreground/background pixel membership, instead of the traditional pixel-wise likelihoods.

- **Chapter 7:** describes how we extend our pixel-wise posterior tracking method to deal with multiple interacting/occluding objects.

- **Part III: System Integration and Conclusions**

  - **Chapter 8:** describes how we use good engineering practice to combine our work on SLAM in dynamic environments and hybrid mapping, with our visual tracking algorithms to build a real system. Part of this system was the design, build and test of a high performance pan-tilt device.
  - **Chapter 9:** concludes with a discussion and suggestions for future work.

1.6 Related Publications

- **[Bibby & Reid, 2005] – Visual Tracking at Sea:** This paper was a result of my 4th year undergraduate work and used a modified version of mean-shift tracking [Comaniciu *et al.*, 2000] to track objects at sea. The unsatisfactory performance of this approach partly motivated this dissertation.

- **[Bibby & Reid, 2006] – Fast Feature Detection with a Graphics Processing Unit Implementation:** This paper presented a novel feature detector, which was comparable to the difference-of-Gaussian method used by [Lowe, 1999], but more suited to a graphics processing unit implementation and hence very fast. This detector is no longer used in any of the methods in the dissertation and therefore not mentioned from here on to maintain clarity.

- **[Bibby & Reid, 2007] – Simultaneous Localisation and Mapping in Dynamic Environments:** This paper describes our work on SLAM in dynamic environments, which will be presented in Chapter 4.

- **[Bibby & Reid, 2008] – Robust Real-Time Visual Tracking using Pixel-Wise Posteriors:** This paper presents our novel method for visual tracking using pixel-wise posteriors, which will be presented in Chapter 6.

- **[Bibby & Reid, 2010a] – A Hybrid SLAM Representation for Dynamic Marine Environments:** This paper presents our hybrid representation for SLAM in marine environments, which will be presented in Chapter 5.

- **[Bibby & Reid, 2010b] – Real-time Tracking of Multiple Occluding Objects using Level Sets:** This paper presents our work on visual tracking of multiple interacting objects using pixel-wise posteriors, which will be presented in Chapter 7.

Chapter 2

Probabilistic Methods

2.1 Introduction

This dissertation shows how *probabilistic methods* can be used to enhance situational awareness in marine environments. The emphasis being on *probabilistic* because the methods used in this dissertation represent their results using probability distributions, as opposed to 'point estimates' or 'best guesses'. Taking this approach adds complexity both computationally and theoretically, but provides a principled way for combining/fusing incoming information received at different times and/or from different sensors. For background reading on the principles behind these probabilistic methods we recommend [Bishop, 2006; Thrun *et al.*, 2005; Manyika & Durrant-Whyte, 1994].

This chapter covers the background knowledge required for the dissertation: Sections 2.2 and 2.3 cover the notation; Section 2.4 covers the probability rules that are used; Section 2.5 introduces graphical models, which we use to represent the probabilistic models; Section 2.6 introduces recursive Bayesian estimation; Section 2.7 introduces the Kalman Filter [Kalman, 1960] (KF); Section 2.8 considers how probability distributions can be represented using a computer and Section 2.9 finishes with a summary.

2.2 General Notation

Throughout this dissertation scalar values will be written in a normal lower-case x whereas vector or multivariate quantities will be written in bold lower-case $\mathbf{x}$.

Matrices are represented using bold upper-case $\mathbf{X}$ and may be given a size e.g. $\mathbf{X}_{3\times3}$ represents a 3-by-3 matrix. The identity matrix is predefined as $\mathbf{I}$ and will often be specified with dimensions. Sets are always defined before use and are bold upper-case, for example $\mathbf{M}_t = \{\mathbf{m}_t^1, \ldots, \mathbf{m}_t^N\}$ defines a set of vector elements $\mathbf{m}_t^1$ through to $\mathbf{m}_t^N$. Sets-of-sets are also used, for example $\mathbf{Y} = \{\mathbf{M}_1, \ldots, \mathbf{M}_T\}$ defines a set of set elements $\mathbf{M}_1$ through to $\mathbf{M}_T$. Subscripts are used to index elements into a set, for example $\mathbf{M}_1$ is the first element of the set $\mathbf{Y}$. Subscripts can also be used to index into a vector, e.g. x_1 is the first element of the vector $\mathbf{x}$, or to index into a matrix e.g. $R_{1,2}$ is the first row and second column of the matrix $\mathbf{R}$. Superscripts are used if another index is required, for example if we are dealing with a set-of-sets then $\mathbf{m}_2^1$ is the first element of the second sub set of $\mathbf{Y}$. Sequences are used to represent a range of values, for example $t:1 = \{t, \ldots, 1\}$; a typical use of a sequence would be to index a set of elements, for example $\mathbf{z}_{t:1} = \{\mathbf{z}_t, \ldots, \mathbf{z}_1\}$.

2.3 Probability Notation

The probability of event X being true will be written as $P(X)$. The probability of $X = x$ will either be written explicitly as $P(X = x)$ or abbreviated for convenience to $P(x)$, as long as it does not confuse meaning. The probability that $X = x$ **given** that $Z = z$ will be written as $P(X = x|Z = z)$ or abbreviated to $P(x|z)$. In a similar fashion, the joint probability that $X = x$ **and** $Z = z$ will be written as $P(X = x, Z = z)$ or abbreviated to $P(x, z)$. For continuous variables $P(x)$ can be interpreted as the probability density and will integrate to one i.e. $\int_x P(x)dx = 1$.

The particular form of distributions used in the text will either be specified in words e.g. a non-parametric RGB histogram with 32 bins per channel or they may be written out explicitly. Parametric Gaussian distributions are often used in the text and are usually written using the following notation:

$$\mathbf{x} \sim N(\tilde{\mathbf{x}}, \boldsymbol{\Sigma}). \tag{2.1}$$

We adopt a slightly different notation, which is more convenient when dealing with least-squares derivations:

$$P(\mathbf{x}) \propto \exp(-\frac{1}{2} \|\tilde{\mathbf{x}} - \mathbf{x}\|^2_{\mathbf{\Sigma}}), \tag{2.2}$$

where: the $\propto$ represents the fact that the normalising constant has been omitted for brevity; the $\tilde{\mathbf{x}}$ represents the mean and the $\|\mathbf{e}\|^2_{\mathbf{\Sigma}}$ notation represents the squared Mahalanobis distance $\mathbf{e}^T\mathbf{\Sigma}^{-1}\mathbf{e}$, where $\mathbf{\Sigma}$ is the covariance.

2.4 Probability Rules

Below is a list of probability rules that are repeatedly applied throughout this dissertation, for a full in context discussion, see [Bishop, 2006; Thrun *et al.*, 2005].

Independence rule: Two probabilities are said to be independent if their joint distribution factors as follows:

$$P(x, z) = P(x)P(z). \tag{2.3}$$

Product rule: This is sometimes referred to as the chain rule and states that:

$$P(x, z) = P(x|z)P(z) = P(z|x)P(x), \tag{2.4}$$

which can also be re-arranged to compute the conditional distributions:

$$P(x|z) = \frac{P(x, z)}{P(z)} \tag{2.5}$$

and

$$P(z|x) = \frac{P(x, z)}{P(x)}. \tag{2.6}$$

Sum rule / marginalisation: This allows a variable to be marginalised (averaged) out from a joint distribution:

$$P(x) = \sum_z P(x, z) = \sum_z P(x|z)P(z). \tag{2.7}$$

Bayes rule: This can be written down directly from the *product rule* and states that:

$$\overbrace{P(x|z)}^{\text{posterior}} = \frac{\overbrace{P(z|x)}^{\text{likelihood}}\overbrace{P(x)}^{\text{prior}}}{\underbrace{P(z)}_{\text{evidence}}}. \tag{2.8}$$

Bayes rule forms the basis for a vast number of useful algorithms, including the ones presented in this dissertation, it is at the heart of methods that coin the phrase 'Bayesian'. The equation is broken into four components: (posterior) this is the distribution that is normally required and represents how the belief in x changes given that z has been observed; (likelihood) is the distribution of z given a particular x; (prior) this is the prior belief of x without any influence of z; (evidence) this is the marginal belief of z without the influence of x.

2.5 Graphical Models

Throughout this dissertation we use graphical models as a design tool to specify our underlying probabilistic model(s). They allow careful design decisions to be made using knowledge about the real-world before having to deal with the mathematics. In particular, they allow one to model the process by which sensor data has been generated.

Figure 2.1 illustrates two simple graphical models that are made up from the following components: circular nodes representing continuous random variables (in this case they are multi-variate variables); arrows representing the conditional relationships between the random variables; shaded nodes representing observed random variables and non-shaded nodes representing hidden random variables.

The joint distribution for a graphical model can written down by visiting each node

(a) (b)

Figure 2.1: (a) Generative model and (b) Bayes rule has been used to do inference on the generative model (the arrow is reversed).

and concatenating a product of $P(parent|children)$, for Figure 2.1(a) this would be:

$$P(\mathbf{x}, \mathbf{z}) = P(\mathbf{z}|\mathbf{x})P(\mathbf{x}) \tag{2.9}$$

and for Figure 2.1(b):

$$P(\mathbf{x}, \mathbf{z}) = P(\mathbf{x}|\mathbf{z})P(\mathbf{z}). \tag{2.10}$$

Equations (2.9) and (2.10) both represent the same joint distribution $P(\mathbf{x}, \mathbf{z})$ and so the graphical models in Figure 2.1 are both valid models for $P(\mathbf{x}, \mathbf{z})$. The difference between Figure 2.1(a) and 2.1(b) becomes apparent when you actually give meaning to the random variables $\mathbf{x}$ and $\mathbf{z}$. If we define $\mathbf{x}$ to be the 2D position of a vehicle and $\mathbf{z}$ to be a measurement of the position, then Figure 2.1(a) can be interpreted as a generative model where the position $\mathbf{x}$ generates/causes the measurement $\mathbf{z}$. Written down mathematically this is Equation (2.9) and contains distributions that we know (or can model) i.e. the likelihood $P(\mathbf{z}|\mathbf{x})$ (sensor model) and the prior $P(\mathbf{x})$. In contrast, Equation (2.10), corresponding to the graphical model in Figure 2.1(b), contains distributions that we cannot easily model i.e. the posterior $P(\mathbf{x}|\mathbf{z})$ (which is actually what we want). By substituting (2.10) into (2.9) and dividing by $P(\mathbf{z}) = \sum_{\mathbf{x}} P(\mathbf{z}|\mathbf{x})P(\mathbf{x})$ we get another form of Bayes rule:

$$P(\mathbf{x}|\mathbf{z}) = \frac{P(\mathbf{z}|\mathbf{x})P(\mathbf{x})}{\sum_{\mathbf{x}} P(\mathbf{z}|\mathbf{x})P(\mathbf{x})}. \tag{2.11}$$

This now gives a distribution of the form $P(\textit{wanted variable(s)}|\textit{measured variable(s)})$, where the *wanted variable(s)* is the 2D position $\mathbf{x}$ and the *measured variable(s)* is the measurement $\mathbf{z}$. Optimisation can be performed on this distribution to compute the Maximum a Posteriori (MAP) estimate. In our example this would be referred to as computing the MAP estimate for the vehicle's 2D location $\mathbf{x}$, given the measurement $\mathbf{z}$. This use of Bayes rule is known as *inference* because the conditional relationship (arrow direction) has been changed so that the distribution $P(\textit{wanted variable(s)}|\textit{measured variable(s)})$ can be inferred from the data. Similar procedures are used throughout this dissertation and can be summarised using the following four steps:

1. Draw a graphical model that represents the process by which measurements are generated, this is known as a generative model.
2. Write down the joint probability distribution by concatenating the product of $P(parent|children)$.
3. Use the probability rules from Section 2.4 to manipulate the joint distribution into a form where $P(\textit{wanted variable(s)}|\textit{measured variable(s)}) = < exp >$.
4. Optionally: Run an optimisation on $< exp >$ to find the MAP estimate.

Note:- Step 4 is occasionally omitted and the distribution from Step 3 is used directly.

These four steps are straightforwardly written down, but can, for a particular problem, require significant time, effort and experience to implement. For instance, many of the problems presented in this dissertation could be modelled using different generative models; selecting a particular one comes down to a combination of experience, trial and error, and making steps 2, 3 and 4 feasible. Once a generative model has been selected and steps 2 and 3 have been applied, it is often still a challenge to perform the optimisation step. The optimisations in this dissertation always involve multivariate variables (sometimes with over a thousand dimensions) and often consist of a mixture of continuous and discrete values. Finding a global optimum may be computationally intractable given the available computing power and so a combination of good engineering practice, approximations and simplifying assumptions is required to produce systems capable of real-time performance.

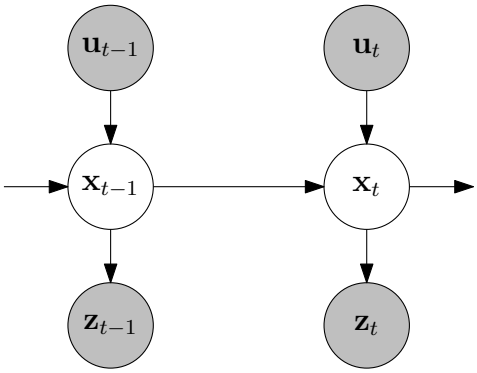


Figure 2.2: Markov process.

2.6 Recursive Bayesian Estimation

Figure 2.2 is a graphical model that represents a Markov process for the hidden variable $\mathbf{x}$ given the observed odometry/control inputs $\mathbf{u}$ and the observed measurements $\mathbf{z}$. The joint distribution can be written down directly from the graphical model:

$$P(\mathbf{x}_{t:0}, \mathbf{z}_{t:1}, \mathbf{u}_{t:1}) = P(\mathbf{x}_0) \prod_{t=1}^{T} P(\mathbf{z}_t|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t). \tag{2.12}$$

The conditional relationships encapsulated within the graphical model (the arrows) imply that this is a Markov process i.e. the system has no memory. This means that the current position only depends on the previous position and the current odometry, which can be written down mathematically as:

$$P(\mathbf{x}_t|\mathbf{x}_{t-1:0}, \mathbf{u}_{t:1}) = P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \tag{2.13}$$

and that the current measurements only depend on the current state

$$P(\mathbf{z}_t|\mathbf{x}_{t:0}) = P(\mathbf{z}_t|\mathbf{x}_t). \tag{2.14}$$

If it is only the current state estimate $\mathbf{x}_t$ that is required, which is often the case in robotics, then this problem can be solved with recursive Bayesian estimation. This

involves recursively applying the following two steps: a prediction step, where the previous state is marginalised out:

$$P(\mathbf{x}_t|\mathbf{z}_{t-1:1}, \mathbf{u}_{t:1}) = \int P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)P(\mathbf{x}_{t-1}|\mathbf{z}_{t-1:1}, \mathbf{u}_{t-1:1})\mathbf{dx}_{t-1} \tag{2.15}$$

and an update step, which incorporates the new measurement using Bayes rule:

$$P(\mathbf{x}_t|\mathbf{z}_{t:1}, \mathbf{u}_{t:1}) = \frac{P(\mathbf{z}_t|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{z}_{t-1:1}, \mathbf{u}_{t:1})}{P(\mathbf{z}_t|\mathbf{z}_{t-1:1}, \mathbf{u}_{t:1})} \tag{2.16}$$

where

$$P(\mathbf{z}_t|\mathbf{z}_{t-1:1}, \mathbf{u}_{t:1}) = \int P(\mathbf{z}_t|\mathbf{x}_t)P(\mathbf{x}_t|\mathbf{z}_{t-1:1}, \mathbf{u}_{t:1})\mathbf{dx}_t. \tag{2.17}$$

2.7 Kalman Filtering

Equations (2.15)-(2.17) give the general form of recursive Bayesian estimation where the probability distributions are left undefined. In the case that the prediction and update probability distributions are modelled as Gaussians and have linear transition models, then Equations (2.15) and (2.16) can be written down in closed form and give the Kalman Filter [Kalman, 1960].

The Kalman Filter is optimal, in the sense that it minimises the mean squared error and is guaranteed to converge if a process exhibits Gaussian noise and linear transition models. However, if the prediction and update probability distributions are modelled as Gaussians but have non-linear transition models then, at the expense of optimality and guaranteed convergence, it is possible to use an extended form of the Kalman Filter, commonly known as the Extended Kalman Filter (EKF). The EKF assumes that the non-linear transition models can be approximated with a linearisation around the current state estimate.

The compromise made when using the EKF is that the linear approximation removes the guarantees of optimality and convergence. If the non-linearities are weak and the linear approximations are good, then the results achieved may also be good. If

however, the underlying functions are highly non-linear then the approximation could be poor. One way to improve the quality of the estimate is to obtain a better linear approximation, which can be achieved by relinearising at the new estimate and then using this improved linearisation to recompute the estimate. This is the principle of the Iterated Extended Kalman Filter (IEKF); relinearisation is computed K times (K is the number of iterations in the IEKF). In practice, this simple extension to the EKF can significantly improve the quality of the estimate, especially if the predicted state is a long way from the true state.

For the Kalman Filtering equations refer to Appendix A and for more detail refer to [Bar-Shalom & Fortmann, 1988; Bar-Shalom *et al.*, 2002].

2.8 Representing Probability Distributions

The biggest implication of using probabilistic methods is that probability distributions are used instead of 'point estimates' or 'best guesses'. Representing these distributions within a computer poses a number of challenges. We will now use a simple example to illustrate the power of probabilistic methods and to consider how probability distributions can be represented within a computer.

A Simple Example

Consider an autonomous vehicle (boat) that has been launched somewhere on a river and has a map of the buoys on the river $\mathbf{M}$ but does not know its location $\mathbf{x}$. The vehicle processes information at discrete time-steps t and uses onboard sensor(s) to make measurement(s) of the colour of any buoys it observes, giving an estimate of where along the river the boat might be $\mathbf{z}_t$. The vehicle is also fitted with a ship's log and can make estimates of how far it has travelled between time-steps, for example $\mathbf{u}_t$ is the distance traveled between time-steps $t-1$ and t. This type of problem is known as Localisation within the robotics community. The graphical model is shown in Figure 2.3 and can be solved using recursive Bayesian estimation, i.e. variations of Equations (2.15) and (2.16).

Figure 2.4 shows a diagram of how the posterior distribution would evolve over three time-steps: the vehicle is initially completely lost and then observes a red buoy, followed by a green buoy and a yellow buoy at time-steps one, two and three respectively.

To initialise the system at time-step zero a prior distribution is used; in this example a uniform distribution[1] is used to represent an equal chance of the vehicle being anywhere along the river. Now the boat moves forwards and at time-step one observes a red buoy $\mathbf{z}_1$ and records the distance on the ship's log $\mathbf{u}_1$. The algorithm applies the prediction (2.15) and the update (2.16) to assign higher probabilities to distances along the river where red buoys exist (four in total). The fact that the algorithm uses a probability distribution to represent the distance $\mathbf{x}$ along the river, means that it is able to model the possibility of the boat being at four different locations. The boat moves again and at time-step two observes a green buoy $\mathbf{z}_2$; the algorithm can incorporate this new information and update the estimate, there are now two locations that are more likely. Finally, the distance $\mathbf{u}_3$ is recorded and a yellow buoy is observed $\mathbf{z}_3$, the estimate is updated and now there is a high probability that the algorithm knows where along the river the boat is.

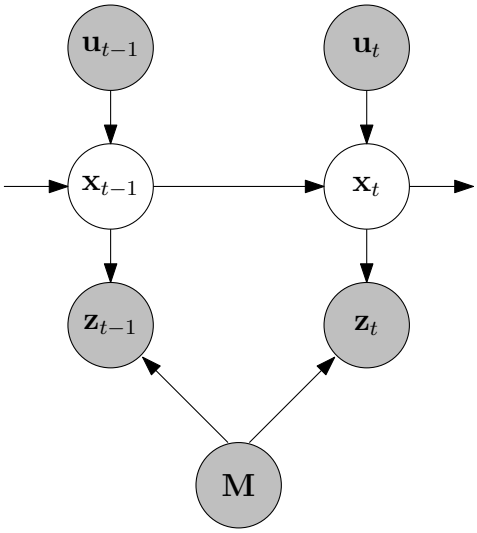


Figure 2.3: A simple example (Localisation).

This example illustrates that by using probability distributions, rather than 'point estimates' or 'best guesses', the algorithm is able to maintain multiple hypotheses and model the uncertainty associated with the variable $\mathbf{x}$. The downside is that whereas a 'point estimate' or 'best guess' is easily represented by a computer, it can be much harder to represent probability distributions. Below is a selection of representations commonly used:

EKF: One of the big downsides of using the EKF for this problem is that it models the probability distributions using a unimodal Gaussian distribution and is therefore

[1]Often referred to as an uninformative distribution.

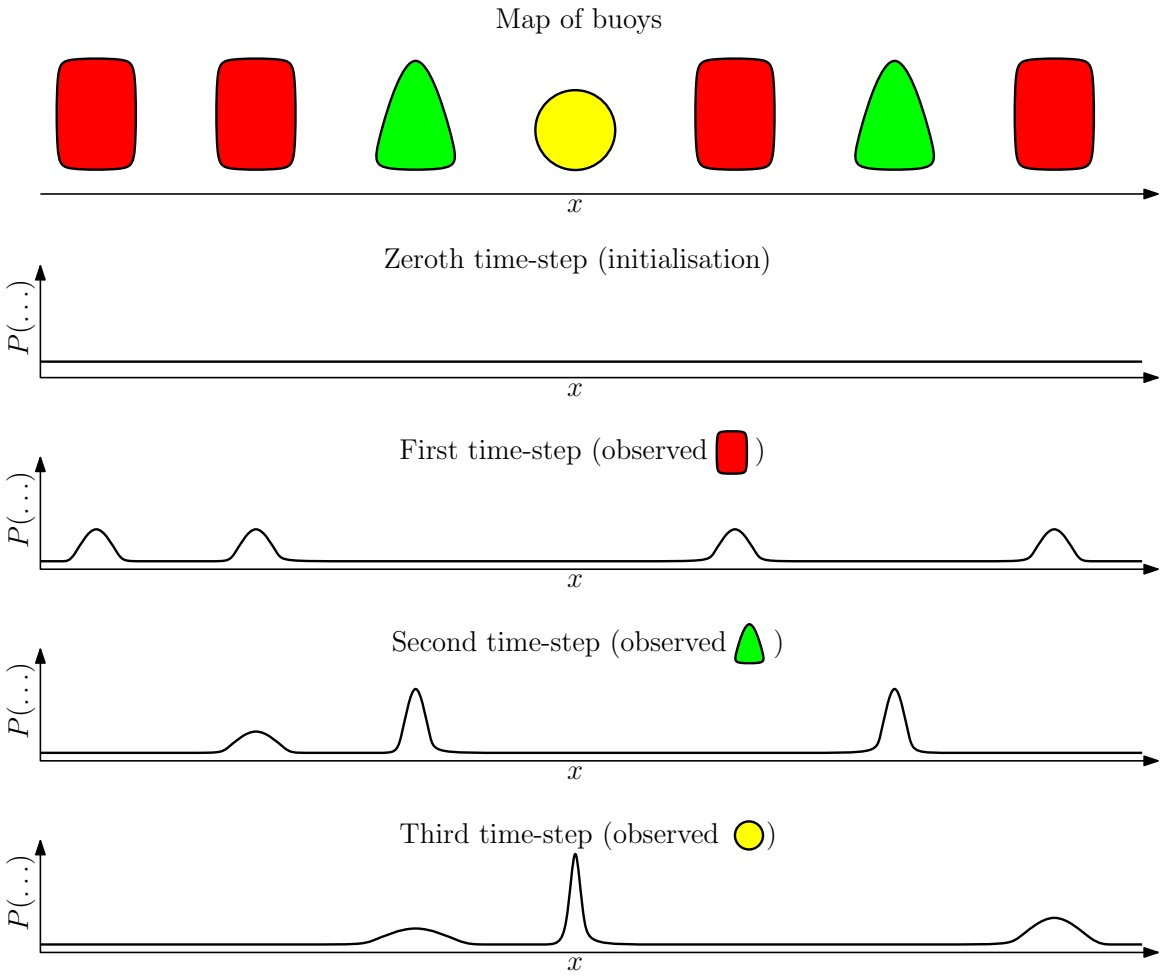


Figure 2.4: An example of a probabilistic approach.

unable to model the possibility that the vehicle could be in several separate locations.

MHT: The unimodal limitations of the EKF can been removed by using a multi-hypothesis EKF or multi-hypothesis tracker (MHT) [Reid, 1979], which allows multiple solutions to be considered. This models the probability distribution as a mixture of Gaussian components. This method would therefore be able to represent our example in Figure 2.4 by having four separate Gaussian components mixed together.

Coarse grid / topological: Unlike the previous two continuous methods, a set of discrete states are placed at interesting locations in the environment and then the vehicle's location $\mathbf{x}$ is treated as a discrete random variable. The probability distribution is then represented using a non-parametric distribution over the discrete states. If applied to our example this approach would use seven discrete states, one for each buoy and the recursive estimation would compute a probability distribution over these discrete states.

Fine grid / metric: Similar to the coarse grid approaches $\mathbf{x}$ is represented using discrete locations and directions; however, a much finer grid is used. This could be applied to our example by discretising the continuous variable $\mathbf{x}$, for example, into one hundred discrete locations. Recursive estimation can then be used to compute a distribution over these discrete states. This approach can handle general distributions but is limited by the resolution of the grid and can be unfeasible for higher dimensional problems.

Monte-Carlo localisation: The work of [Isard & Blake, 1998a] proposed a different way to approximate probability distributions based on a set of weighted particles. These particles and their weights are distributed such that their local density approximates the density of the underlying probability distribution. [Isard & Blake, 1998a] applied their idea to visual tracking and showed that they could achieve higher levels of robustness compared with the unimodal Kalman Filter based approaches. This could be applied to the example in Figure 2.4 and would use a number of particles to capture the multiple modes present in the probability distribution.

In summary, there are many different ways of representing probability distributions within the computer, each of which has different strengths and weaknesses. The EKF for example is easy to implement and runs efficiently because the model comprises a single Gaussian. If the true distribution is modelled well by a single Gaussian then the EKF can do a good job; if however, the true distribution is non Gaussian, like our example in Figure 2.4, then the EKF can perform very poorly. At the other end of the scale are the Monte-Carlo based methods, which are able to model arbitrary probability distributions and can therefore handle problems such as the one in Figure 2.4. The downside is that Monte-Carlo methods suffer from the curse of dimensionality and the number of particles required grows exponentially with the number of dimensions modelled. Although work has been carried out to make more efficient use of a given number of particles [Isard & Blake, 1998b; Montemerlo *et al.*, 2003] and to adapt the number of particles required [Fox, 2001]; unfortunately, the number is often still too large for real-time performance.

In this dissertation we use a variety of different representations, including: uni-modal Gaussians, multi-modal Gaussians, discrete non-parametric distributions and fine grid metric representations.

2.9 Summary

This chapter has introduced the notation, probability rules, graphical models, recursive Bayesian estimation and how to represent probability distributions using a computer. These basic tools will be used throughout this dissertation, both when estimating the metric information using our SLAM in dynamic environments and when doing visual tracking using our pixel-wise posterior methods. We will now move onto the first part of the dissertation, which looks at how we can estimate metric information in the marine environment using sensors mounted on mobile vehicles.

Part I

Estimating Metric Information

We will now look at how to estimate metric information using sensors mounted on a mobile vehicle. In our practical experiments this vehicle is a boat; however, the methods and theory presented in this part of the dissertation could be applied to a wide variety of vehicles and sensors. At the heart of our solutions is SLAM [Smith et al.*, 1988], which has been studied intensively in the robotics community since the late 1980s. The idea behind SLAM is to use sensor(s) attached to a mobile vehicle to automatically build a map of the environment and to simultaneously use that map to localise the vehicle's own position. The sensors used often include some combination of: camera(s), laser range finder(s), SONAR sensor(s) and RADAR sensor(s). From a theoretical standpoint, SLAM uses probabilistic methods to fuse incoming sensor data and infer both the position of the vehicle and the positions of any objects in the surrounding environment.*

Chapter 3

Simultaneous Localisation and Mapping

3.1 Introduction

This chapter covers the background on SLAM that is required to understand and implement our work on SLAM in dynamic environments in Chapter 4 and hybrid mapping in Chapter 5. It begins in Section 3.2 by introducing the common notation. Then Sections 3.3 and 3.4 look at two simpler problems: (localisation) which estimates a vehicle's location given a map of the environment and (mapping) which estimates a map of the environment given accurate positions of the vehicle. This is extended in Section 3.5 to the more general case of an unknown map and location, which requires SLAM [Smith *et al.*, 1988]. Sections 3.6, 3.7 and 3.8 describe three different techniques, respectively, for solving the SLAM problem: (i) the recursive filtering solution; (ii) Full-SLAM / Bundle Adjustment and (iii) Sliding Window SLAM, which is the approach taken in this dissertation and captures the best aspects of recursive filtering and Full-SLAM. Section 3.8 looks at the structure of the information matrix and describes how marginalisation is performed in information form and Section 3.10 finishes with a summary.

3.2 Notation

Below is a summary of the notation used in this chapter:

- $\mathbf{x}_t$: A state vector describing the vehicle's pose at time t.
- $\mathbf{u}_t$: A control vector that was applied to vehicle at time $t-1$ to take it to time t.
- $\mathbf{z}_t$: A measurement made by the vehicle at time t of a landmark in the world.
- $\mathbf{m}_k$: A state vector describing the location of landmark k.
- $\mathbf{X} = \{\mathbf{x}_0, \ldots, \mathbf{x}_t\}$: A set of vehicle poses.
- $\mathbf{U} = \{\mathbf{u}_1, \ldots, \mathbf{u}_t\}$: A set of odometry.
- $\mathbf{Z} = \{\mathbf{z}_1, \ldots, \mathbf{z}_t\}$: A set of measurements.
- $\mathbf{M} = \{\mathbf{m}_0, \ldots, \mathbf{m}_k\}$: A set of all landmarks.

3.3 Localisation

Localisation involves estimating a mobile vehicle's location $\mathbf{x}$ using a map $\mathbf{M}$ of the environment that is provided a priori, given measurements $\mathbf{Z}$. The level of uncertainty in the initial position can vary from being completely lost, to knowing where the vehicle was t seconds ago and having some idea of how it has moved since then. The example used in Section 2.8 from the previous chapter was a simple example of the localisation problem.

Depending on the level of uncertainty, different approaches for localisation can be taken, using different probabilistic representations and estimation techniques. For a review of early work on localisation refer to [Borenstein *et al.*, 1996]. One early solution to the problem was developed by [Leonard & Durrant-Whyte, 1991] and used artificial beacons to provide the map that the vehicle can then localise itself to. This type of approach is very practical as the artificial beacons can be placed or tagged in such a way that data-association is unambiguous, making the posterior a unimodal distribution and therefore 'easy' to solve (for example using an EKF). This is essentially how the GPS system works. Numerous approaches have been taken for solving harder problems where multi-modal distributions are required. For instance, [Jensfelt & Kristensen, 1999; Roumeliotis & Bekey, 2000] apply multi-hypothesis tracking (MHT) [Reid, 1979], which allows multiple solutions to be considered. The work of [Simmons & Koenig, 1995] used a coarse grid/topological representation applied to

office environments where the discrete states were placed at corridor junctions and encoded four discrete directions. A finer grid has been used by [Burgard *et al.*, 1996] where grid cells are 15cm × 15cm and encode 180 directions for a 4m × 7m room. Monte-Carlo methods using particle filtering [Isard & Blake, 1998a] have been applied to localisation by [Dellaert *et al.*, 1999].

3.4 Mapping

Mapping is the complement of Localisation; the objective is to estimate a map $\mathbf{M}$ of the surrounding environment, given that you accurately know the location of the vehicle $\mathbf{x}$ when it makes an observation of the environment $\mathbf{z}$ using onboard sensor(s). Figure 3.1 shows the graphical model that corresponds to the mapping problem, where the key difference from localisation (shown in Figure 2.3) is that $\mathbf{x}$ is now observed and $\mathbf{M}$ is unobserved; note that the motion model has been removed because $\mathbf{x}$ is now given. Mapping is applicable if you have a method for measuring the absolute position of the vehicle e.g. GPS and a compass.

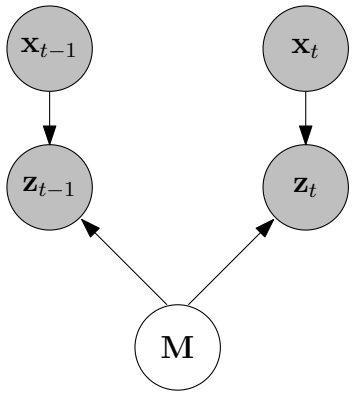


Figure 3.1: Mapping

Mapping in the context of mobile robotics has also been studied for several decades. For a good summary of the early work on mapping refer to [Borenstein *et al.*, 1996]. A common approach to mapping is to use occupancy grids, which partition the world into discrete grid cells and then estimate the probability that a particular grid cell is occupied. Much of the original work on occupancy grids was down to [Elfes, 1987] and [Moravec, 1988]. The objective of occupancy grid mapping is to compute the posterior $P(\mathbf{M}|\mathbf{x}_{t:1}, \mathbf{z}_{t:1})$, where $\mathbf{M}$ is the map consisting of N grid cells $\mathbf{M} = \{m_1, \ldots, m_N\}$. Unfortunately, the number of possible solutions to $\mathbf{M}$ grows exponentially with N

e.g. for a 100×100 grid there would be 2^{10000} combinations and so it has become standard practice to make the simplifying assumption that neighbouring grid cells are independent of one another i.e.

$$P(\mathbf{M}|\mathbf{x}_{t:1}, \mathbf{z}_{t:1}) = \prod_{n=1}^{N} P(m_n|\mathbf{x}_{t:1}, \mathbf{z}_{t:1}). \tag{3.1}$$

The occupancy grid can then be updated by modeling the inverse sensor model $P(m_n|\mathbf{x}_{t:1}, \mathbf{z}_{t:1})$ and updating the grid cells independently, see [Thrun *et al.*, 2005] for full details. A more sophisticated method for computing occupancy grids using the full posterior $P(\mathbf{M}|\mathbf{x}_{t:1}, \mathbf{z}_{t:1})$ was proposed by [Thrun, 2002] but it requires batch processing of the data and is not suitable for incremental updating. In Chapter 5 our hybrid representation uses an occupancy grid. We assume independent grid cells because we need to be able to incrementally update the grid.

3.5 SLAM

SLAM is a generalisation of localisation and mapping to the case where both the vehicle's trajectory $\mathbf{x}$ and the map $\mathbf{M}$ are unknown. Figure 3.2 shows a graphical model representing the SLAM problem, where both the $\mathbf{x}$ nodes and the $\mathbf{M}$ node are now hidden random variables. The dotted line between $\mathbf{x}_{t-1}$ and $\mathbf{M}$ represents the fact that there may be correlations between the vehicle pose and the map if marginalisation (a prediction step) has been performed.

3.6 EKF SLAM

The first stochastic formulation of the SLAM problem was first proposed in [Smith *et al.*, 1988] and used an EKF to do the recursive Bayesian estimation. The concept is to augment the vehicle state $\mathbf{x}^v$ (where the vehicle is) with the states of any landmarks $\mathbf{x}^m$ (the map):

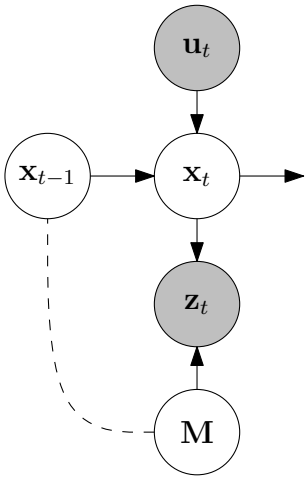


Figure 3.2: SLAM

$$\mathbf{x}^{'} = \begin{bmatrix} \mathbf{x}^{v} \\ \mathbf{x}_1^{m} \\ \cdots \\ \mathbf{x}_N^{m} \end{bmatrix}, \tag{3.2}$$

where N is the number of landmarks in the map. Likewise, the covariance $\mathbf{P}$ is also augmented:

$$\mathbf{P}^{'} = \begin{bmatrix} \mathbf{P}_{vv} & \mathbf{P}_{vm} \\ \mathbf{P}_{vm}^{T} & \mathbf{P}_{mm} \end{bmatrix}, \tag{3.3}$$

where $\mathbf{P}_{vv}$ is the covariance of the vehicle, $\mathbf{P}_{mm}$ is the covariance of the map and $\mathbf{P}_{vm}$ is the correlations between the map and the vehicle. The prediction distribution is of the same form as Equation (A.10), with the difference that the state transition Jacobians $\boldsymbol{\Delta}\mathbf{F}_x^{'}$ and $\boldsymbol{\Delta}\mathbf{F}_u^{'}$ now have the form:

$$\boldsymbol{\Delta}\mathbf{F}_x^{'} = \begin{bmatrix} \boldsymbol{\Delta}\mathbf{F}_x & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{2n\times 2n} \end{bmatrix} \tag{3.4}$$

and

$$\mathbf{\Delta F}_u^{'} = \begin{bmatrix} \mathbf{\Delta F}_u & \mathbf{0} \\ \mathbf{0} & \mathbf{I}_{2n\times 2n} \end{bmatrix}, \tag{3.5}$$

where $\mathbf{\Delta F}_x$ and $\mathbf{\Delta F}_u$ are the Jacobians before state augmentation; and the prediction covariance has the form:

$$\mathbf{Q}^{'} = \begin{bmatrix} \mathbf{Q} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix}. \tag{3.6}$$

This has the effect of applying the same motion model to the vehicle as would be in pure localisation and applies a constant position, zero noise, motion model to the landmarks (the map). This is the correct assumption if the landmarks are truly stationary; if however, any of the landmarks should move then this assumption will lead to map corruption (we deal with this problem in Chapter 4). The update distribution has the same form as Equation (A.11) except that the measurement model Jacobian is now long and thin:

$$\mathbf{H}^{'} = \begin{bmatrix} \mathbf{\Delta H}_{\mathbf{x}^v} & \dots & \mathbf{\Delta H}_{\mathbf{x}_d^m} & \dots \end{bmatrix}, \tag{3.7}$$

where $\mathbf{\Delta H}_{\mathbf{x}^v}$ is the Jacobian with respect to the vehicle, $\mathbf{\Delta H}_{\mathbf{x}_d^m}$ is the Jacobian with respect to the measured landmark and the data-association d selects the correct landmark. Finally, the last modification is the method for adding a new landmark, which uses the function $\mathbf{x}_{N+1}^m = g(\mathbf{x}^v, \mathbf{z})$ to predict the new landmark location and the Jacobian:

$$\mathbf{\Delta G}_x = \begin{bmatrix} \mathbf{I}_{N\times N} & \mathbf{0} \\ [\mathbf{\Delta G}_{x^v}\mathbf{0}] & \mathbf{\Delta G}_z \end{bmatrix} \tag{3.8}$$

of $g(\mathbf{x}^v, \mathbf{z})$ to initialise the new covariance:

$$\mathbf{P}'^{+} = \boldsymbol{\Delta}\mathbf{G}_x \begin{bmatrix} \mathbf{P}' & \mathbf{0} \\ \mathbf{0} & \mathbf{R} \end{bmatrix} \boldsymbol{\Delta}\mathbf{G}_x^T. \tag{3.9}$$

This process of adding new landmarks is added into the EKF algorithm (see Appendix A) after the update step, which leads to the Extended Kalman Filter SLAM (EKF-SLAM) algorithm.

The EKF-SLAM algorithm has been in common use since the original [Smith *et al.*, 1988] paper. One of the first applications to real-world problems was due to [Leonard & Durrant-Whyte, 1991], who used artificial landmarks as beacons. Using the augmented state representation introduces correlations between the vehicle and the map $\mathbf{P}_{vm}$ and between landmarks and other landmarks $\mathbf{P}_{mm}$. [Csorba, 1997] discussed the importance of these correlations and showed how they are essential for consistent estimation. [Csorba, 1997] also looked into the convergence properties of SLAM and showed that the map covariance would never increase. A more critical look at the consistency of the EKF-SLAM algorithm can be found in [Castellanos *et al.*, 2004]. In particular, they use the Normalised Estimation Error Squared (NEES) (see [Bar-Shalom *et al.*, 2002]) to analyse the consistency of EKF-SLAM. We base some of our analysis in Chapter 4 on this methodology.

[Leonard & Rikoski, 2001] show how decisions can be delayed within the EKF-SLAM framework. They delay marginalising out old vehicle positions until they are no longer involved in any active decisions, this has the effect of maintaining correlations between the most recent pose and older poses. These correlations can then be used to add constraints that involve more than one pose. This is particularly useful in the example where you have a bearing only sensor and cannot initialise a new landmark immediately; instead it is necessary to wait until the landmark has been observed over a significant baseline and the depth can be estimated. This approach is similar to the fixed-lag Kalman Smoother [Anderson & Moore, 1979]; however, [Leonard & Rikoski, 2001] do not necessarily marginalise out old poses in the order that they were added. It should also be pointed out that this has close relationships with the work of [Sibley *et al.*, 2007] on Sliding Window SLAM (see Section 3.8) and with our work in Chapters 4 and 5.

[Dissanayake *et al.*, 2001, 2002] introduce the idea of map management; in particular, they show that a significant number of landmarks can be removed from the estimation

whilst remaining consistent. The process for deleting a landmark is straightforward for EKF-SLAM and involves deleting the rows and columns corresponding to the landmark from the state vector and the covariance matrix. Deleting landmarks from the EKF in this way is equivalent to marginalising out the uncertainty in that landmark.

One of the downsides of EKF-SLAM is its quadratic computational complexity, due to the inversion of the $N \times N$ covariance matrix. In practice, this limits the standard EKF-SLAM system to map sizes of around one hundred landmarks. Significant effort has gone into pushing this boundary, starting with the work of [Leonard & Feder, 1999] who proposed the idea of decoupled stochastic mapping or sub-mapping as it has become known. Considerable research effort has continued in the area of sub-mapping [Guivant & Nebot, 2001; Williams, 2001; Leonard & Feder, 2001; Tardos *et al.*, 2002]. These methods do not overcome the quadratic complexity, but improve the scaling of EKF-SLAM so that it can be applied to larger problems i.e. thousands of landmarks.

Information Filters

The information filter can be considered the dual of the covariance filter (EKF), rather than an estimated state vector $\mathbf{x}$ and covariance matrix $\mathbf{P}$, an information vector $\mathbf{y}$ and information matrix $\mathbf{Y}$ are maintained. The relationship between the information filter and the covariance filter is:

$$\begin{aligned} \mathbf{Y} &= \mathbf{P}^{-1} \\ \mathbf{y} &= \mathbf{P}^{-1}\mathbf{x}. \end{aligned} \tag{3.10}$$

In other words, the information matrix is the inverse of the covariance matrix and the information vector is the state vector projected using the information matrix. There have been numerous approaches to SLAM that use this information form, for example [Csorba, 1997; Newman & Durrant-Whyte, 2001; Eustice *et al.*, 2005; Dellaert & Kaess, 2006; Sibley *et al.*, 2007]. There are also many approaches, which use the information form to reduce the complexity of the problem. [Thrun *et al.*, 2004] proposed the sparse extended information filter, which essentially removes very weak links in the information filter making it more efficient to solve. [Paskin, 2003] proposed an efficient factorisation of the posterior using thin junction trees, again

to improve efficiency. Throughout this dissertation, our SLAM algorithms use the information form of the SLAM problem to do the underlying estimation, this will be discussed further in the following two sections.

3.7 Bundle Adjustment / Full SLAM

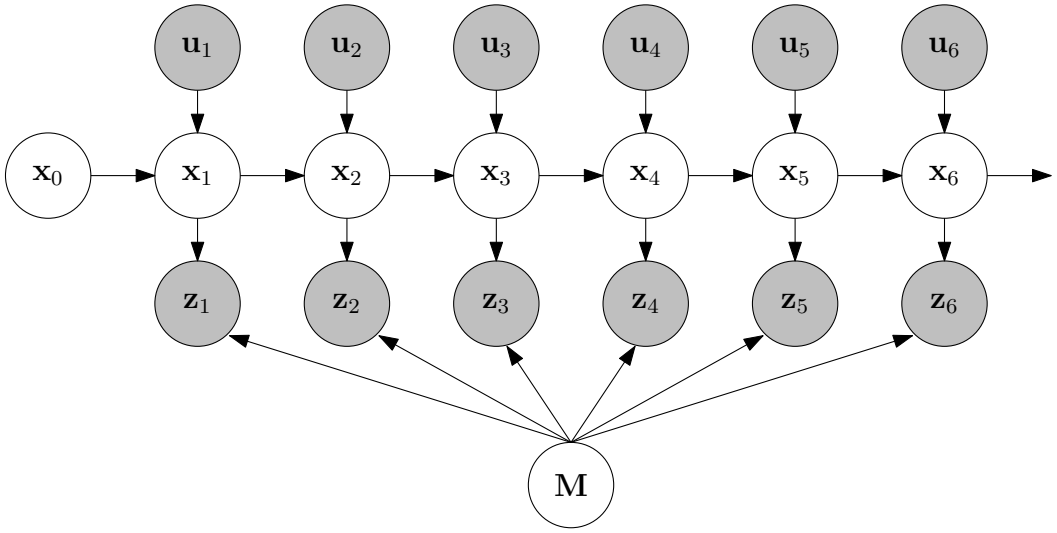


Figure 3.3: Full SLAM

Each iteration of "standard" EKF-based SLAM provides a MAP estimate for the state at the current time-step. Modern understanding of this recognises that previous poses have been marginalised out. In contrast, Full-SLAM [Thrun *et al.*, 2005; Dellaert & Kaess, 2006] finds the MAP estimate of the entire pose history. This is akin to bundle-adjustment techniques from photogrammetry [Triggs *et al.*, 2000; Hartley & Zisserman, 2004], and has the advantage that more accurate solutions can be found since optimisation is performed over past and future data. It does of course suffer from the problem of growth without bound in the state size. Figure 3.3 shows the graphical model corresponding to a Full SLAM solution, where the entire trajectory (seven poses) are estimated together.

The next section will introduce a Sliding Window SLAM technique proposed by [Sibley *et al.*, 2007] that captures benefits of Full-SLAM without the unbounded growth in state size. First, however, let us review the least-squares derivation that forms the basis for both Full-SLAM [Thrun *et al.*, 2005; Dellaert & Kaess, 2006] and Sliding Window SLAM [Sibley *et al.*, 2007].

We will make two simplifying assumptions: (i) only one observation per time-step and (ii) known data-association i.e. which landmark generated a given measurement (we will relax these assumptions in Chapter 4). The joint probability of $\mathbf{X}$, $\mathbf{M}$, $\mathbf{U}$ and $\mathbf{Z}$ can be factorised, using the conditional structure depicted by the graphical model in Figure 3.3, as follows:

$$P(\mathbf{X}, \mathbf{M}, \mathbf{U}, \mathbf{Z}) = P(\mathbf{x}_0)P(\mathbf{M})\prod_{t=1}^{T} P(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M})P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t), \tag{3.11}$$

where:

- T is the number of time-steps.
- $P(\mathbf{x}_0)$ is the prior on the vehicle state, which has a mean $\tilde{\mathbf{x}}_0$ and covariance $\mathbf{P}_0$.
- $P(\mathbf{M})$ is the prior on the map, which is normally taken to be the uninformative uniform distribution.
- $P(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M})$ is the measurement model i.e. the probability of the measurement $\mathbf{z}_t$ given the vehicle pose $\mathbf{x}_t$, the map $\mathbf{M}$ and the correct data-association.
- $P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$ is the motion model i.e. the probability of the new pose $\mathbf{x}_t$ given the last vehicle pose $\mathbf{x}_{t-1}$ and the odometry $\mathbf{u}_t$.

If we take $P(\mathbf{M})$ to be the uninformative uniform distribution then (3.11) reduces to:

$$P(\mathbf{X}, \mathbf{M}, \mathbf{U}, \mathbf{Z}) = P(\mathbf{x}_0)\prod_{t=1}^{T} P(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M})P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t). \tag{3.12}$$

Let us now also make Gaussian assumptions and define the prior term, motion model and measurement model respectively as:

$$\mathbf{x}_0 = \tilde{\mathbf{x}}_0 + \mathbf{p}_0 \quad \Leftrightarrow \quad P(\mathbf{x}_0) \propto exp(-\frac{1}{2}\left\|\tilde{\mathbf{x}}_0 - \mathbf{x}_0\right\|^2_{\mathbf{P}_0}) \tag{3.13}$$

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}, \mathbf{u}_t) + \mathbf{q}_t \quad \Leftrightarrow \quad P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \propto exp(-\frac{1}{2}\left\|f(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t\right\|^2_{\mathbf{Q}_t}) \tag{3.14}$$

$$\mathbf{z}_t = h(\mathbf{x}_t, \mathbf{M}) + \mathbf{r}_t \quad \Leftrightarrow \quad P(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M}) \propto exp(-\frac{1}{2}\left\|h(\mathbf{x}_t, \mathbf{M}) - \mathbf{z}_t\right\|^2_{\mathbf{R}_t}) \tag{3.15}$$

where $\mathbf{p}_0$, $\mathbf{q}_t$ and $\mathbf{r}_t$ are normally distributed, zero mean, noise vectors with covariances $\mathbf{P}_0$, $\mathbf{Q}_t$ and $\mathbf{R}_t$ respectively. We can now perform inference on the graphical model in Figure 3.3 to find the MAP estimate $\{\hat{\mathbf{X}}, \hat{\mathbf{M}}\} = \arg\max_{\{\mathbf{X},\mathbf{M}\}} P(\mathbf{X}, \mathbf{M}|\mathbf{U}, \mathbf{Z})$. This can be done by minimising the negative log of the joint distribution (3.12):

$$\{\hat{\mathbf{X}}, \hat{\mathbf{M}}\} \triangleq \underset{\{\mathbf{X},\mathbf{M}\}}{\arg\min} \left(-\log(P(\mathbf{X}, \mathbf{M}, \mathbf{U}, \mathbf{Z}))\right). \tag{3.16}$$

By substituting Equations (3.13), (3.14) and (3.15) into (3.16) we get a non-linear least-squares problem of the form:

$$\{\hat{\mathbf{X}}, \hat{\mathbf{M}}\} \triangleq \underset{\{\mathbf{X},\mathbf{M}\}}{\arg\min} \Big\{ \left\|\tilde{\mathbf{x}}_0 - \mathbf{x}_0\right\|^2_{\mathbf{P}_0} + \sum_{t=1}^{T} \big(\left\|f(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t\right\|^2_{\mathbf{Q}_t} + \left\|h(\mathbf{x}_t, \mathbf{M}) - \mathbf{z}_t\right\|^2_{\mathbf{R}_t} \big) \Big\}. \tag{3.17}$$

Let us now linearise the non-linear terms and re-write as a matrix equation:

$$\{\hat{\mathbf{X}}, \hat{\mathbf{M}}\} \triangleq \underset{\{\mathbf{X},\mathbf{M}\}}{\arg\min} \Big\{ \left\|-\boldsymbol{\delta}\mathbf{x}_0 - \{\mathbf{x}_0 - \tilde{\mathbf{x}}_0\}\right\|^2_{\mathbf{P}_0} + \sum_{t=1}^{T} \big(\left\|\{\mathbf{F}_{t-1}\boldsymbol{\delta}\mathbf{x}_{t-1} - \boldsymbol{\delta}\mathbf{x}_t\} - \{\mathbf{x}_t - f(\mathbf{x}_{t-1}, \mathbf{u}_t)\}\right\|^2_{\mathbf{Q}_t} + \left\|\{\mathbf{H}_t\boldsymbol{\delta}\mathbf{x}_t + \mathbf{J}_t\boldsymbol{\delta}\mathbf{M}\} - \{\mathbf{z}_t - h(\mathbf{x}_t, \mathbf{M})\}\right\|^2_{\mathbf{R}_t} \big) \Big\}, \tag{3.18}$$

where $\mathbf{F}_{t-1}$ is the Jacobian of $f(.)$ with respect to $\mathbf{x}_{t-1}$, $\mathbf{H}_t$ is the Jacobian $h(.)$ with respect to $\mathbf{x}_t$ and $\mathbf{J}_t$ is the Jacobian of $h(.)$ with respect to $\mathbf{M}$. We can now factorise

and write a standard least-squares matrix equation:

$$\mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{A}\boldsymbol{\delta} = \mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{b}, \tag{3.19}$$

where $\mathbf{A}$ is a matrix of Jacobians, $\boldsymbol{\Sigma}$ is a covariance matrix, $\mathbf{b}$ is an error vector and $\boldsymbol{\delta}$ is the correction vector for the state. For a simple example illustrating the structure of $\mathbf{A}$, $\boldsymbol{\Sigma}^{-1}$, $\mathbf{b}$ and $\boldsymbol{\delta}$, refer to Appendix B, or for a more detailed explanation refer to [Dellaert & Kaess, 2006]. The matrix $\mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{A}$ is known as the Hessian matrix and for SLAM problems where $\mathbf{X}$ and $\mathbf{M}$ are observable it will form a positive definite matrix i.e. $\mathbf{v}^T\mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{A}\mathbf{v} > 0$ for all non-zero vectors $\mathbf{v}$. This means that $\boldsymbol{\delta}$ can be solved for using a Cholesky decomposition of the Hessian and a back substitution. The state is then updated as follows:

$$\{\mathbf{X}, \mathbf{M}\} = \{\mathbf{X}, \mathbf{M}\} + \boldsymbol{\delta}, \tag{3.20}$$

and the Jacobians forming the matrix $\mathbf{A}$ are recomputed, the process is then repeated until the solution converges. The Cholesky decomposition and back solve can either be achieved using standard dense matrix arithmetic or by using direct sparse methods [Davis, 2006; Dellaert & Kaess, 2006]. We have implementations for both approaches, but find in general that optimised dense matrix approaches are faster for the size of problems we deal with. If however, the problem size is above five to six hundred states then the overhead of the sparse methods is worth considering.

3.8 Sliding Window SLAM

Sliding Window SLAM [Sibley *et al.*, 2007] is essentially a compromise between traditional filtering solutions (e.g. EKF-SLAM) and Full-SLAM. The idea is that marginalisation is postponed for $t - \tau$ time-steps leaving a temporal sliding window during which pose estimation is computed. This effectively allows the benefits of a Full-SLAM solution to be obtained for states that lie within the temporal sliding window. In our work we take advantage of the optimisation over the trajectory history not only to improve the pose estimates, but crucially in order to allow reversible data-association and model-selection to take place. Full details of this will be presented

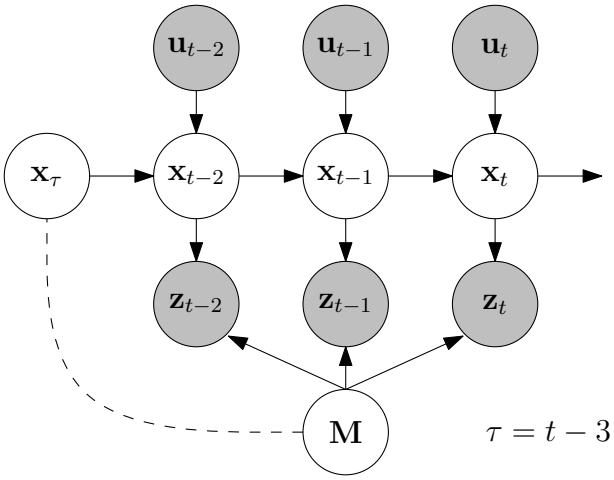


Figure 3.4: Sliding Window SLAM (τ is the beginning of the sliding window).

in Chapters 4 and 5, but first we will review the two governing equations of sliding window SLAM which are: (i) an optimisation step

$$\{\hat{\mathbf{x}}_{\tau:T}, \hat{\mathbf{M}}\} = \underset{\{\mathbf{x}_{\tau:T}, \mathbf{M}\}}{\arg\max} \left(P(\mathbf{x}_\tau, \mathbf{M}|\mathbf{z}_{1:\tau}, \mathbf{u}_{1:\tau}) \prod_{t=\tau+1}^{T} P(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M}) P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \right) \quad (3.21)$$

and (ii) a marginalisation step

$$P(\mathbf{x}_{\tau+1}, \mathbf{M}|\mathbf{z}_{1:\tau+1}, \mathbf{u}_{1:\tau+1}) = \int P(\mathbf{x}_{\tau+1}, \mathbf{x}_\tau, \mathbf{M}|\mathbf{z}_{1:\tau+1}, \mathbf{u}_{1:\tau+1}) \mathbf{d}\mathbf{x}_\tau. \quad (3.22)$$

The term $P(\mathbf{x}_\tau, \mathbf{M}|\mathbf{z}_{1:\tau}, \mathbf{u}_{1:\tau})$ is the prior used at time T and is just the posterior at time τ i.e. the distribution of vehicle pose and map at the beginning of the temporal sliding window. In practice this is only recalculated when $\tau > 0$, before that time the distribution of initial vehicle pose $P(\mathbf{x}_0)$ is used and the marginalisation step is left out.

The Best of Both Worlds

It is interesting to consider what happens to the sliding window method if: (i) the window length only covers a single pose i.e. $\tau = T - 1$ and (ii) the window length covers all time i.e. $\tau = 0$. Let us consider the first case, if we substitute $\tau = T - 1$

into Equations (3.21) and (3.22) then we get:

$$\{\hat{\mathbf{x}}_{T-1:T}, \hat{\mathbf{M}}\} = \underset{\{\mathbf{x}_{T-1:T}, \mathbf{M}\}}{\arg\max} \left(P(\mathbf{x}_{T-1}, \mathbf{M}|\mathbf{Z}_{1:T-1}, \mathbf{U}_{1:T-1}) P(\mathbf{z}_T|\mathbf{x}_T, \mathbf{M}) P(\mathbf{x}_T|\mathbf{x}_{T-1}, \mathbf{u}_T)\right) \tag{3.23}$$

and

$$P(\mathbf{x}_T, \mathbf{M}|\mathbf{Z}_{1:T}, \mathbf{U}_{1:T}) = \int P(\mathbf{x}_T, \mathbf{x}_{T-1}, \mathbf{M}|\mathbf{Z}_{1:T}, \mathbf{U}_{1:T}) \mathbf{dx}_{T-1}, \tag{3.24}$$

which is equivalent to the IEKF [Bar-Shalom & Fortmann, 1988; Bar-Shalom *et al.*, 2002]. Let us now consider what happens if $\tau = 0$ (note that $\tau = 0$ and that if $\tau \leq 0$ then $P(\mathbf{x}_0)$ is used for the prior and the marginalisation is left out):

$$\{\hat{\mathbf{x}}_{0:T}, \hat{\mathbf{M}}\} = \underset{\{\mathbf{x}_{0:T}, \mathbf{M}\}}{\arg\max} \left(P(\mathbf{x}_0) \prod_{t=1}^{T} P(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M}) P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \right), \tag{3.25}$$

which is nothing other than Full-SLAM/Bundle Adjustment. This shows that the sliding window estimator really does give us the best of both worlds; by selecting the length of the temporal sliding window we can go from computing the IEKF solution for a single pose and the map all the way to the Full-SLAM solution over all time-steps [Thrun *et al.*, 2005].

3.9 The Information Matrix

We will now consider the relationship between the Hessian in the least-squares formulation ($\mathbf{A}^T\mathbf{\Sigma}^{-1}\mathbf{A}$ in (3.19)), the information matrix in the information formulation (3.10) and the covariance matrix in the covariance formulation (3.3). In statistics, the Cramer Rao lower bound [Scharf & McWhorter, 1993] states that the inverse of the Fisher Information matrix $Y(\mathbf{x})$ gives a lower bound for the covariance matrix of an unbiased estimator $\hat{\mathbf{x}}$ i.e.

$$\text{cov}_{\hat{\mathbf{x}}}(\hat{\mathbf{x}}) \geq Y(\mathbf{x})^{-1}, \tag{3.26}$$

where $\mathbf{A} \geq \mathbf{B}$ means that $\mathbf{A} - \mathbf{B}$ is positive semidefinite. Therefore if the Cramer Rao lower bound is reached, then given that we have normally distributed zero mean variables the following condition is satisfied:

$$\mathbf{P}^{-1} = \mathbf{Y} = \mathbf{A}^T \mathbf{\Sigma}^{-1} \mathbf{A}$$

where $\mathbf{A}^T \mathbf{\Sigma}^{-1} \mathbf{A}$ is the approximation to the Hessian calculated when solving the least-squares problem (3.19). This is why the information matrix $\mathbf{Y}$ in the information filter version of SLAM [Eustice *et al.*, 2005] and the Hessian in the least-squares formulation of SLAM [Sibley *et al.*, 2007] are equivalent to the inverse of the covariance matrix $\mathbf{P}$ in the more traditional Kalman filter based SLAM systems. Interestingly, as explained fully in [Dellaert & Kaess, 2006], the non-zero elements of the information matrix $\mathbf{Y}$ correspond to links in the Markov Random Field (MRF) that is equivalent to the graphical model in Figure 3.3. Each of these links represent a constraint or relationship between two nodes in the MRF, e.g. a measurement equation linking a vehicle pose to a landmark or an odometry equation linking one vehicle pose to the next. Figure 3.5 shows the structure of the information matrix and MRF for a simple 2D example: $\mathbf{Y}_v$ is block tridiagonal and represents the information from odometry between vehicle poses; $\mathbf{Y}_m$ is block diagonal and represents the information about landmarks in the map and $\mathbf{Y}_{vm}$ and $\mathbf{Y}_{vm}^T$ represent the information associated with measuring a landmark from a given pose.

Marginalisation in Information Form

Equation (3.22) is the marginalisation of the oldest pose in the temporal sliding window. Given that we are using a least-squares formulation i.e. a uni-modal Gaussian distribution, it is possible to do the marginalisation analytically in information form. It is well known that it is possible to decouple states $\mathbf{y}_1$ from a system of equations of the form:

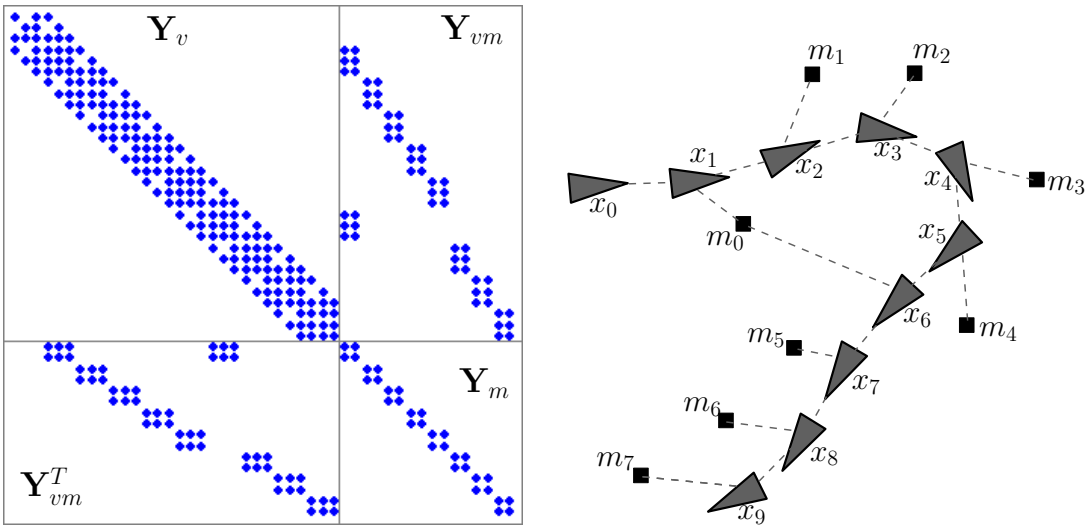


Figure 3.5: Shows the structure of the information matrix (left column) and the corresponding MRF (right column) for a simple 2D example. The triangles represent vehicle poses, the squares represent landmark locations (these triangles and squares are the nodes of the MRF) and the dotted lines represent constraints, which correspond to non zero elements in the information matrix.

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{B}^T & \mathbf{D} \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 \end{bmatrix} \tag{3.27}$$

using the Schur Complement method (see [Triggs *et al.*, 2000] for details). The idea is to pre-multiply both sides of the equation with the matrix $[\mathbf{I} \quad \mathbf{0}; -\mathbf{B}^T\mathbf{A}^{-1} \quad \mathbf{I}]$, which results in a system of equations where $\mathbf{y}_2$ can be solved independently of $\mathbf{y}_1$ i.e.

$$\begin{bmatrix} \mathbf{A} & \mathbf{B} \\ \mathbf{0} & \mathbf{D} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B} \end{bmatrix} \begin{bmatrix} \mathbf{y}_1 \\ \mathbf{y}_2 \end{bmatrix} = \begin{bmatrix} \mathbf{b}_1 \\ \mathbf{b}_2 - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{b}_1 \end{bmatrix}. \tag{3.28}$$

The term $\mathbf{D} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$ is known as the Schur Complement and corresponds to the information matrix for the decoupled system. If this system of equations represents a least-squares problem as described in Section 3.7, then this is equivalent to marginalising out the random variables $\mathbf{y}_1$. Let us now consider what happens to the structure of $\mathbf{D} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$ as old poses are marginalised out. Figure 3.6 shows the effect of marginalising out poses one-by-one. The first row shows the situation before any marginalisation. The second row corresponds to marginalising out $\mathbf{x}_0$, which results

in no change of structure in the information matrix (because no features were observed from this pose) but does introduce a prior on the vehicle state $\mathbf{x}_1$. Then in the third row $\mathbf{x}_1$ has been marginalised out and a link has been introduced between $\mathbf{x}_2$ and $\mathbf{m}_0$, which is also seen in the $\mathbf{Y}_{vm}$ block of the information matrix (this extra link and the prior on $\mathbf{x}_2$ and $\mathbf{m}_0$ is explained by the prior term in Equation (3.21)). As poses $\mathbf{x}_2$ and $\mathbf{x}_3$ are marginalised out more links are again introduced between the oldest pose and the landmarks; links are also introduced between landmarks that are no longer observed from the oldest pose. In practice, we use the prior term (3.13) in our least-squares formulation to represent the prior term (3.22) in the sliding window. *To maintain probabilistic correctness only equations containing the pose being marginalised out should be included in the system* (3.27) *and then the modified system* (3.28) *should be solved for* $\mathbf{y}_2$.

In practice, we store the information matrix that corresponds to the prior term (3.22) separately. This prior information matrix has non-zero elements corresponding to the poses, landmarks and links that have been shaded grey in Figure 3.6. We then add in the information corresponding to the measurement equations and odometry equations that are associated with the poses being optimised in the temporal sliding window.

3.10 Summary

This chapter has introduced SLAM both in covariance form and information form. It has shown how SLAM can be solved recursively using the EKF or in batch form using non-linear least-squares. A temporal sliding window framework was introduced, which can be used as a compromise between the recursive methods and the batch methods whilst maintaining real-time performance. In the next chapter, we will show how the use of a temporal sliding window allows us to do reversible decision making and how this can be used to achieve more robust data-association and to include dynamic objects in the SLAM estimation.

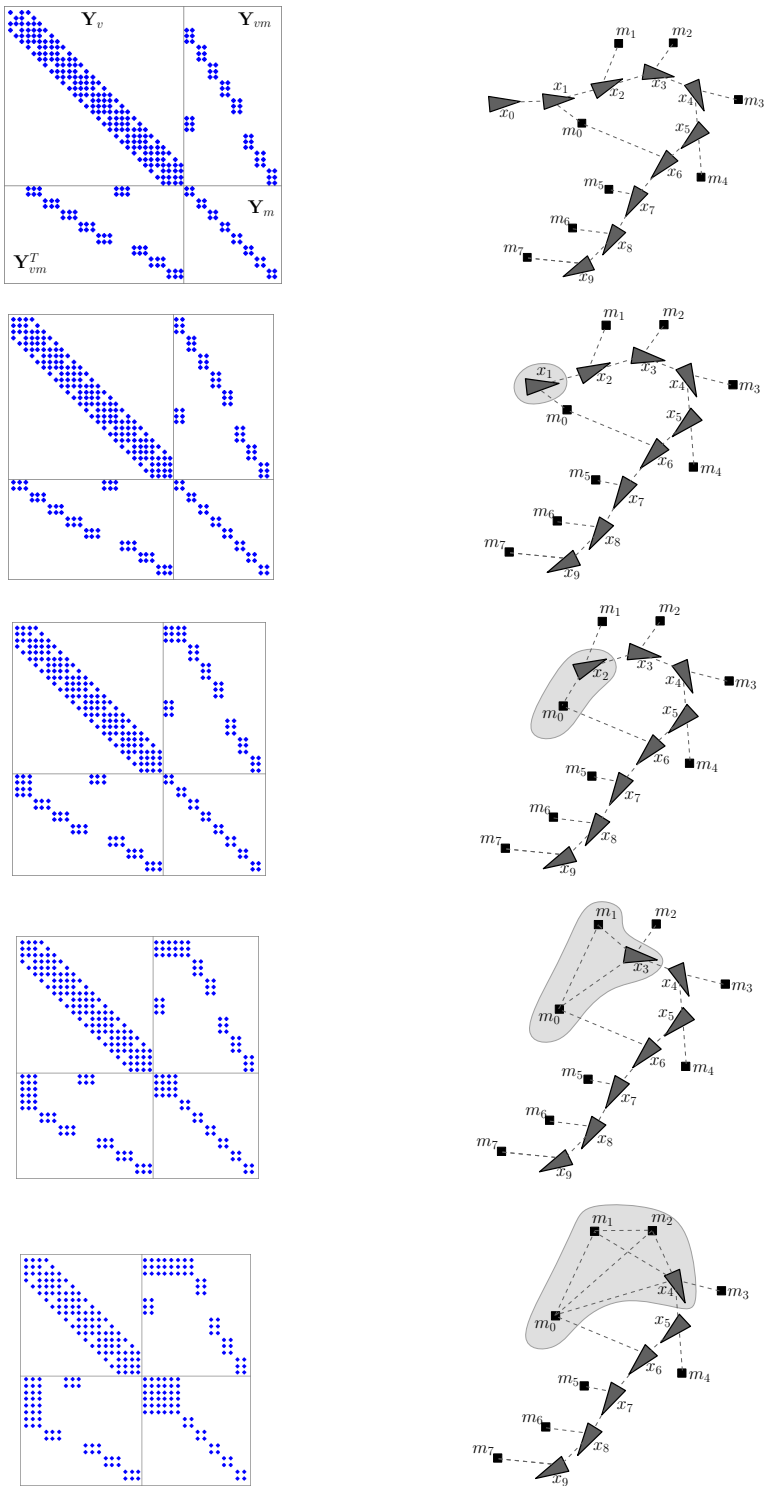


Figure 3.6: Illustrates how the information matrix (left column) and MRF (right column) change as poses $\mathbf{x}_0$ (row 2), $\mathbf{x}_1$ (row 3), $\mathbf{x}_2$ (row 4) and $\mathbf{x}_3$ (row 5) are marginalised out. The light grey region indicates the nodes that are involved in the prior term (3.22) in Equation (3.21).

Chapter 4

SLAM in Dynamic Environments

4.1 Introduction

We have now covered the necessary background on probabilistic methods and SLAM and will now introduce the first contribution of the dissertation, which is how to include dynamic objects directly in a SLAM system.

Previous chapters have made the assumption that landmarks are always stationary and eluded to the problems of inconsistency if in fact some of them are dynamic. [Wang *et al.*, 2003] have tackled this problem by detecting moving objects in the environment and tracking them separately with EKFs. [Hähnel *et al.*, 2003] use expectation maximisation to compute a posterior for each scan element as to whether it is stationary or dynamic. Having successfully classified dynamic scan elements, they remove them before performing localisation and mapping and hence obtain improved localisation and better maps. They demonstrate their system in environments with large portions of dynamic data e.g. busy corridors. [Wolf & Sukhatme, 2004] keep two complementary maps one for stationary aspects of the environment and one for dynamic aspects. These maps are represented using occupancy grids (see Section 3.4) and enforce mutual exclusion, in other words a grid cell can only be stationary or dynamic, but not both. They demonstrate good results and show that by excluding dynamic grid cells they are able to build better maps. In contrast to these methods, we include dynamic objects directly within the augmented SLAM representation. This means that our method is not only keeping correspondences between dynamic objects over time, but it also uses the correlation in the SLAM filter to help the estimate of the dynamic objects' and the vehicle's trajectory.

In order to include dynamic objects directly in the SLAM representation we must be able to solve the model-selection problem. The estimator constantly needs to answer the question: is a landmark moving or is it stationary? Although there are methods for doing model-selection in recursive filtering frameworks, such as interacting multiple model estimation or generalised pseudo-Bayesian estimation [Bar-Shalom *et al.*, 2002], these methods always have some lag before the model-selection parameter(s) converge to the correct steady state. This means that for a period of time the filter could classify a target as dynamic when it is stationary or vice-versa. This is potentially catastrophic for SLAM, because incorrectly modeling a dynamic or stationary landmark will lead to biased measurements and hence map corruption and inconsistency.

We propose a framework that combines sliding window SLAM [Sibley *et al.*, 2007] (as described in Section 3.8) and generalised expectation maximisation [Neal & Hinton, 1998]. This allows us to include reversible model-selection and data-association parameters in the estimation and hence include dynamic objects in the SLAM map robustly. The key to our method is the use of a temporal sliding window optimisation, which delays the point when information is marginalised out (3.22), allowing the filter a period of time to get the model-selection and data-association parameters correct before marginalisation. *Although something similar could be achieved with a more traditional EKF using delayed decision making [Leonard & Rikoski, 2001], the difference is that our method uses reversible as opposed to delayed decision making i.e. decisions can change many times in light of new information before being committed to the estimate.*

The adverse effects of poor data-association in SLAM, namely inconsistent estimates and divergence, are normally unacceptable and hence a suitable method must be selected. A common approach is the chi-squared Nearest Neighbour (NN) test, which assumes independence between landmarks and then probabilistically chooses the best measurement which falls within the gate of an individual landmark. This method can work well for sparsely distributed environments with good sensors; however, once the proximity between landmarks approaches the sensor noise, or clutter is present, ambiguous situations arise and a more sophisticated method is required. One such method is Joint Compatibility Branch and Bound (JCBB) [Neira & Tardos, 2001], which takes into account the correlations between landmarks by searching an interpretation tree [Grimson, 1990] for the maximum number of jointly compatible associations. This method produces very good results when ambiguities are present,

but still suffers from problems in the presence of clutter and is slow for large numbers of measurements. More recently, the data-association problem has been treated as a discrete optimisation over multiple time-steps [Wijesoma *et al.*, 2006]. We also treat data-association as a discrete optimisation, but include model-selection and propose an alternative method that uses sliding window optimisation and generalised expectation maximisation [Neal & Hinton, 1998](more specifically an approximate method called classification expectation maximisation [Celeux & Govaert, 1992]).

This chapter will begin in Section 4.2 by describing our method for doing SLAM with reversible data-association; Section 4.3 extends this to SLAM in dynamic environments; Section 4.4 compares our methods to an IEKF with either NN or JCBB for data-association and demonstrates the system working on real-world data and finally Section 4.5 finishes with some concluding remarks.

4.2 Reversible Data-Association

The previous chapters have covered the necessary background knowledge, let us now introduce the first of the algorithms presented in this dissertation. We first relax our assumption of known data-association and introduce integer data-association parameters $\mathbf{D} \triangleq \{d_1, \ldots, d_t\}$, which assign measurement $\mathbf{z}_t$ to landmark $\mathbf{m}_{d_t}$. By combining sliding window estimation and least-squares with generalised expectation maximisation, we can estimate both the continuous state estimates $\{\hat{\mathbf{X}}, \hat{\mathbf{M}}\}$ and the discrete data-association parameters $\mathbf{D}$.

Figure 4.1 illustrates the graphical model that corresponds to the joint distribution of the relaxed problem:

$$P(\mathbf{X}, \mathbf{M}, \mathbf{D}, \mathbf{U}, \mathbf{Z}) = P(\mathbf{x}_0)P(\mathbf{M})P(\mathbf{D})\prod_{t=1}^{T} P(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M}, d_t)P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t). \qquad (4.1)$$

What we are really after is the MAP estimate $P(\mathbf{X}, \mathbf{M}|\mathbf{U}, \mathbf{Z})$, whereas what we have is $P(\mathbf{X}, \mathbf{M}, \mathbf{D}, \mathbf{U}, \mathbf{Z})$ where $\mathbf{D}$ is considered a nuisance parameter. The Bayesian approach for dealing with this is to use the sum rule to marginalise out $\mathbf{D}$ i.e.

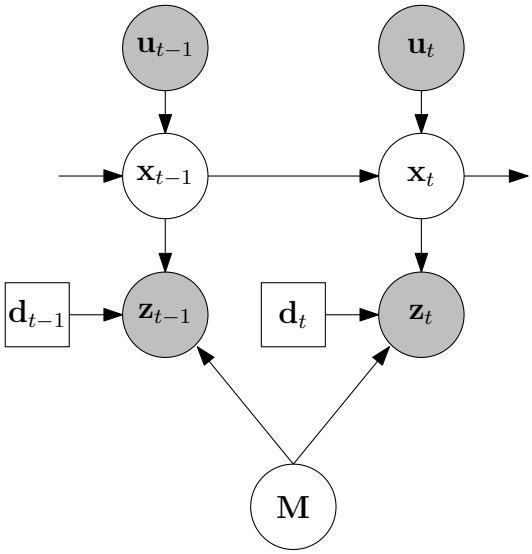


Figure 4.1: A graphical model representing SLAM with reversible data-association (note:- square boxes indicate discrete variables).

$$P(\mathbf{X}, \mathbf{M}|\mathbf{U}, \mathbf{Z}) = \sum_{\mathbf{D}} P(\mathbf{X}, \mathbf{M}, \mathbf{D}|\mathbf{U}, \mathbf{Z}). \tag{4.2}$$

Unfortunately, in practice this summation is computationally intractable because the number of permutations of $\mathbf{D}$ grows exponentially with the length of the temporal sliding window. A more tractable solution is to use the expectation maximisation algorithm [Dempster *et al.*, 1977; Dellaert, 2002] to estimate $P(\mathbf{X}, \mathbf{M}|\mathbf{U}, \mathbf{Z})$. If we let $\mathbf{\Theta} = \{\mathbf{X}, \mathbf{M}\}$ and $\mathbf{\Psi} = \{\mathbf{U}, \mathbf{Z}\}$ then we would like $P(\mathbf{\Theta}|\mathbf{\Psi})$ as opposed to $P(\mathbf{\Theta}, \mathbf{D}|\mathbf{\Psi})$. The expectation maximisation algorithm achieves this by recursively applying the following two steps:

- E-Step: Calculate $P(\mathbf{D}|\mathbf{\Theta}^k, \mathbf{\Psi})$.
- M-Step: $\mathbf{\Theta}^{k+1} = \arg\max_{\mathbf{\Theta}} \left(\sum_{\mathbf{D}} P(\mathbf{D}|\mathbf{\Theta}^k, \mathbf{\Psi}) \log P(\mathbf{\Theta}|\mathbf{D}, \mathbf{\Psi}) \right)$.

A simplification that is often applied to make the M-Step even more tractable is the 'winner-take-all' approach, also known as classification expectation maximisation [Celeux & Govaert, 1992; Meila & Heckerman, 1998], which assumes $P(\mathbf{D}|\mathbf{\Theta}^k, \mathbf{\Psi})$ to be a delta function centred on the best value of $\mathbf{D}$, reducing the algorithm to:

- E-Step: $\mathbf{D}^{k+1} = \arg\max_{\mathbf{D}} P(\mathbf{D}|\mathbf{\Theta}^k, \mathbf{\Psi})$.

- M-Step: $\boldsymbol{\Theta}^{k+1} = \arg\max_{\boldsymbol{\Theta}} P(\boldsymbol{\Theta}|\mathbf{D}^{k+1}, \boldsymbol{\Psi})$.

Finally, it has been shown that it is not necessary to complete the maximisation, but that a single step where $P(\boldsymbol{\Theta}^{k+1}|\mathbf{D}^{k+1}, \boldsymbol{\Psi}) >= P(\boldsymbol{\Theta}^{k}|\mathbf{D}^{k}, \boldsymbol{\Psi})$ is not only sufficient for convergence but often improves the rate of convergence [Neal & Hinton, 1998]. For probabilistic correctness it is necessary to use the joint distribution over landmarks and a single pose during the E-Step i.e. JCBB. In practice this is very slow for large numbers of measurements and so we also include in our results an implementation which makes an extra assumption of landmark independence during the E-Step i.e. chi-squared NN. In practice this method gives a significant improvement over other methods (which do not use reversible data-association) without the full cost of JCBB. It is also interesting at this point to draw on the similarity between this approach and iterative closest point [Besl & Mckay, 1992]; the significant difference is that our method uses the underlying probability distribution (Mahalanobis distances) to find the most likely correspondences, as opposed to the closest in a euclidean sense. Algorithm 1 gives a summary of our method.

Algorithm 1: SLAM with reversible data-association.

$\mathbf{P} = \mathbf{P}_0; \mathbf{x}_0 = \tilde{\mathbf{x}}_0; \mathbf{M} = []; \mathbf{D} = [];$
for *t=[0:T]* **do**
  DoVehiclePrediction();
  while $|\boldsymbol{\delta}|_\infty > \epsilon$ **do**
    $\hat{\mathbf{D}}$ = DoDataAssociation();
    AddAnyNewLandmarks();
    Compute $\mathbf{A}$, $\boldsymbol{\Sigma}$ and $\mathbf{b}$;
    Solve for $\boldsymbol{\delta}$ in $\mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{A}\boldsymbol{\delta} = \mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{b}$;
    $\{\hat{\mathbf{x}}_{\tau:t}, \hat{\mathbf{M}}\} = \{\hat{\mathbf{x}}_{\tau:t}, \hat{\mathbf{M}}\} + \boldsymbol{\delta}$;
    Compute the covariance matrix $\mathbf{P}$;
  end
  if $\tau > 0$ **then**
    Compute $\mathbf{y}_2$ and $\mathbf{D} - \mathbf{B}^T\mathbf{A}^{-1}\mathbf{B}$ using Schur Complement method (see Section 3.9);
  end
end

4.3 SLAM in Dynamic Environments

We will now introduce our method for SLAM in Dynamic Environments (SLAMIDE). Let us start by relaxing the problem even further by: (i) introducing model-selection

parameters $\mathbf{V}_T \triangleq \{v_T^0, \ldots, v_T^k\}$, which consist of a binary indicator variable per landmark taking the value **stationary** or **dynamic** with probability p, $1-p$ respectively and (ii) extending the state vector for each landmark to include velocities $\dot{\mathbf{x}}$. Figure 4.2 is a Bayesian network that shows our formulation of the SLAMIDE problem, where the most significant changes from normal SLAM are:

- The map becomes time dependent $\mathbf{M} \triangleq \{\mathbf{M}_0, \ldots, \mathbf{M}_t\}$.
- Model-selection parameters $\mathbf{V}_T \triangleq \{v_T^0, \ldots, v_T^k\}$ are introduced.
- Data-association parameters $\mathbf{D} \triangleq \{d_1, \ldots, d_t\}$ are introduced.

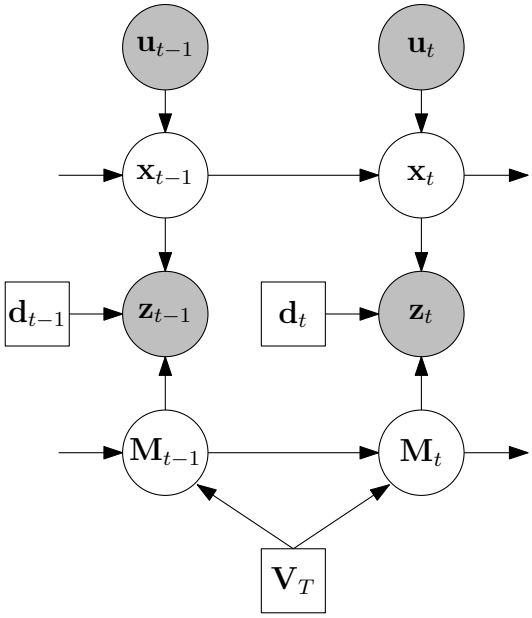


Figure 4.2: A graphical model representing SLAMIDE (note:- square boxes indicate discrete variables).

The corresponding joint distribution $P(\mathbf{X}, \mathbf{M}, \mathbf{D}, \mathbf{V}, \mathbf{U}, \mathbf{Z})$ from Figure 4.2 is:

$$P(\mathbf{X}, \mathbf{M}, \mathbf{D}, \mathbf{V}, \mathbf{U}, \mathbf{Z}) = P(\mathbf{x}_0)P(\mathbf{M}_0)P(\mathbf{D})P(\mathbf{V}_T)\times \prod_{t=1}^{T} P(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M}_t, d_t)P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)P(\mathbf{M}_t|\mathbf{M}_{t-1}, \mathbf{V}_T), \quad (4.3)$$

where:

- $P(\mathbf{V}_T)$ is a prior on the model-selection parameters.
- $P(\mathbf{M}_t|\mathbf{M}_{t-1}, \mathbf{V}_T)$ is the motion model for the map given the current estimate of the model-selection parameters. We use constant position for stationary landmarks and constant velocity with noise in $\dot{\mathbf{x}}$ for dynamic landmarks.

Following the same principles used in our reversible data-association method and including the extra nuisance parameter $\mathbf{V}$ we propose the following five steps to solve the optimisation:

1. $\mathbf{D}^{k+1} = \arg\max_{\mathbf{D}} P(\mathbf{D}|\boldsymbol{\Theta}^k, \mathbf{V}^k, \boldsymbol{\Psi})$
2. $\boldsymbol{\Theta}^{k+1} = \arg\max_{\boldsymbol{\Theta}} P(\boldsymbol{\Theta}|\mathbf{D}^{k+1}, \mathbf{V}^k, \boldsymbol{\Psi})$
3. $\mathbf{M}^{k+1'} = \arg\max_{\mathbf{M}} P(\mathbf{M}|\mathbf{X}^{k+1}, \mathbf{D}^{k+1}, \mathbf{V} = \mathbf{dyn}, \boldsymbol{\Psi})$
4. $\mathbf{V}^{k+1} = \arg\max_{\mathbf{V}} P(\mathbf{V}|\mathbf{X}^{k+1}, \mathbf{M}^{k+1'}, \mathbf{D}^{k+1}, \boldsymbol{\Psi})$
5. $\boldsymbol{\Theta}^{k+1} = \arg\max_{\boldsymbol{\Theta}} P(\boldsymbol{\Theta}|\mathbf{D}^{k+1}, \mathbf{V}^{k+1}, \boldsymbol{\Psi})$

Step 1: performs the data-association using either NN or JCBB. In practice this is actually also computed at every iteration in Steps 2, 3 and 5.

Step 2: is a least-squares optimisation for the vehicle poses and landmark states using the new data-association. The main purpose of this optimisation is to refine the predicted vehicle and landmark locations using the new measurements. In practice this step is particularly important if the vehicle prediction is poor (large odometry noise), because large vehicle uncertainty gives rise to an ambiguous situation where it is hard to differentiate between vehicle and landmark motion.

Step 3: optimises for the landmark states assuming all landmarks are dynamic whilst holding the vehicle poses constant. The reason the vehicle poses are held constant is to remove any ambiguity between vehicle and landmark motion. This is reasonable if most of the landmarks maintain their model-selection between time-steps and hence the $\mathbf{X}^{k+1}$ given by Step 2 is close to the optimal answer.

Step 4: takes the answer from Step 3 and computes the MAP estimate for $\mathbf{V}_T$ using a recursive Bayesian filter; where the likelihood model is a Gaussian on the average velocity with σ=2.0m/s and μ=0 and the prior $P(\mathbf{V}_T)$ for landmark j is:

$$P(v_T^j = \textbf{stationary}) = \begin{cases} 0.6 & \text{if } v_{T-1}^j = \textbf{stationary}, \\ 0.4 & \text{if } v_{T-1}^j = \textbf{dynamic}, \end{cases}$$

which is based on $\mathbf{V}_{T-1}$ the model-selection parameters chosen at the last time-step.

Step 5: is a least-squares optimisation with the new model-selection and data-association parameters (using the answer from Step 2 as the starting point for optimisation). This step refines the estimate from Step 2 taking into account any changes in model-selection to give the final estimate for this time-step.

In practice this whole process only requires a few least-squares iterations, typically: two in Step 2; one in Step 3 and two or three in Step 5; Step 1 and Step 4 are solved directly. See Algorithm 2 for a summary in pseudo-code.

Map management: When adding a new landmark we initialise its model-selection probability to 0.5 (to reflect the uncertainty in whether it is dynamic or stationary) and add a very weak prior of zero initial velocity. This weak prior is essential to make sure that a landmark's velocity is always observable and hence our system of equations is positive definite. We also remove any dynamic landmarks that are not observed within the temporal sliding window, this is done for two reasons: (i) real-world objects do not obey a motion model exactly and so errors accumulate if you predict for too long and (ii) if you continue predicting a dynamic landmark and hence adding noise, then at some point measurements begin to get incorrectly associated to it due to the Mahalanobis test.

4.4 Results

We use two simple 2D environments, which cover 400m by 400m, one with 15 landmarks and the other with 20 landmarks. In both environments the vehicle moves between three waypoints at 5m/s using proportional heading control (maximum yaw rate 5°/sec) and provides rate-of-turn, forwards velocity and slip as odometry with covariance $\mathbf{Q}$. It has a range bearing sensor with a 360° field-of-view, 400m range and zero mean Gaussian noise added with covariance $\mathbf{R}$. The second environment is used for the dynamic object experiment where we progressively change stationary landmarks to dynamic landmarks, which move between waypoints using the same

Algorithm 2: SLAMIDE

$\Pi = \mathbf{P}_0^{-1}; \mathbf{P} = \mathbf{P}_0; \tilde{\mathbf{x}} = \mathbf{x}_0; \hat{\mathbf{X}} = \mathbf{x}_0;$
$\mathbf{M} = []; \mathbf{D} = []; \mathbf{V} = [];$
for *t=[0:T]* **do**
  DoVehiclePrediction();
  V=**dyn**;
  DoMapPrediction();
  while $|\boldsymbol{\delta}|_\infty > \epsilon_1$ **do**
    $\hat{\mathbf{D}}$ = DoDataAssociation();
    Compute $\mathbf{A}$, $\boldsymbol{\Sigma}$ and $\mathbf{b}$;
    Solve for $\boldsymbol{\delta}$ in $\mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{A}\boldsymbol{\delta} = \mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{b}$;
    $\{\hat{\mathbf{X}}_{\tau:t}^{k+1}, \hat{\mathbf{M}}^{k+1}\} = \{\hat{\mathbf{X}}_{\tau:t}^{k}, \hat{\mathbf{M}}^{k}\} + \boldsymbol{\delta}$;
    Compute the covariance matrix $\mathbf{P}$;
  end
  V=DoModelSelection();
  while $|\boldsymbol{\delta}|_\infty > \epsilon_2$ **do**
    $\hat{\mathbf{D}}$ = DoDataAssociation();
    AddAnyNewLandmarks();
    Compute $\mathbf{A}$, $\boldsymbol{\Sigma}$ and $\mathbf{b}$;
    Solve for $\boldsymbol{\delta}$ in $\mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{A}\boldsymbol{\delta} = \mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{b}$;
    $\{\hat{\mathbf{X}}_{\tau:t}^{k+1}, \hat{\mathbf{M}}^{k+1}\} = \{\hat{\mathbf{X}}_{\tau:t}^{k}, \hat{\mathbf{M}}^{k}\} + \boldsymbol{\delta}$;
    Compute the covariance matrix $\mathbf{P}$;
  end
  if $\tau > 0$ **then**
    Compute $\tilde{\mathbf{x}}$ and $\boldsymbol{\Pi}$ using Schur Complement method (see Section 3.9);
  end
end

control scheme, speed and rate-of-turn as the vehicle.

We compare our method using either NN or JCBB for data-association against an IEKF with either NN or JCBB. All experiments are 60 time-steps long and use a chi-squared threshold of $\boldsymbol{\upsilon}^T\mathbf{S}^{-1}\boldsymbol{\upsilon} < 16$, a temporal sliding window length of 6 time-steps, odometry noise (all noise quotes are for 1σ) of 0.1m/s on forwards velocity and 0.01m/s on slip, measurement noise of 1m for range and 0.5° for bearing, a maximum number of 8 iterations, the same stopping condition and the same initial vehicle uncertainty. The reason we use such a large chi-squared threshold is because for any significant angular uncertainty linearising the prediction covariance can cause all measurements to fall outside their data-association gates.

In order to compare the performance of the algorithms we use two metrics: (i) the percentage of correct data-association, which we define to be the percentage of correct associations between time-steps with respect to the total number of potential correct associations and (ii) the percentage of consistent runs. We use the following test to determine whether a run is consistent – compute the Normalised Estimation Error Squared (NEES), see [Bar-Shalom *et al.*, 2002], which is defined as $D_t^2 = (\mathbf{x}_t - \hat{\mathbf{x}}_t)^T\mathbf{P}_t^{-1}(\mathbf{x}_t - \hat{\mathbf{x}}_t)$ and then for each time-step perform the corresponding chi-squared test $D_t^2 \leq \chi^2_{r,1-\alpha}$ where r is the dimension of $\mathbf{x}_t$ and α is a threshold (which we take to be 0.05). The probability of this test failing k times out of n can be computed from a binomial distribution, which we use to threshold on the number of times the test can fail before we are 99% certain that a run is inconsistent.

We have carried out three Monte-Carlo simulation experiments (where each point on the graphs has been generated from 100 runs):

Figure 4.3 - Noise in rate-of-turn odometry: Performance was tested without clutter against increasing noise in rate-of-turn odometry from a one sigma of 1° to 60°. The IEKFJCBB and our RDJCBB both perform perfectly with data-association but start becoming inconsistent more often for higher noise levels. As expected our RDNN outperforms the IEKFNN and matches the performance of IEKFJCBB up to around 25° of noise; this is interesting because it shows the RDNN could be used as a faster alternative to IEKFJCBB for medium noise problems.

Figure 4.4 - Number of clutter measurements: Performance was tested with a noise of 1° for rate-of-turn odometry against increasing clutter from 0 to 100 clutter measurements within the sensor range. This is where the real benefit of reversible data-association becomes apparent. All algorithms tested use the same map manage-

ment scheme, which is to remove landmarks that are not observed for three consecutive time-steps after they have been added to the map. In the traditional IEKF this is done by simply removing them from the state vector and covariance matrix (marginalisation); whereas with our scheme if the information is removed before marginalisation i.e. the temporal sliding window is longer than the time required to carry out map management then there is no effect on the estimate. This is clear from Figure 4.4 as both of our methods maintain their consistency with increasing clutter as opposed to the IEKF based methods which tail off.

Figure 4.5 - Percentage of dynamic objects: Performance was tested without clutter and with a noise of 1° for rate-of-turn odometry against an increasing percentage of dynamic objects from 0 to 100 percent. The figure clearly shows that using SLAMIDE to include dynamic objects allows us to navigate in regions with dynamic objects. We maintain a good level of consistency for up to 90% of dynamic objects at which point the performance degrades until at 100% every run is inconsistent, which is because the system is no longer observable i.e. there are ambiguities between vehicle and landmark motion.

Timing results: With 20 measurements per time-step and a temporal sliding window length of 6 time-steps on a 3.6GHz Pentium 4 the IEKF and SLAM with reversible data-association run at approximately 30Hz and SLAMIDE runs at about 3Hz. We believe this can be significantly improved upon as we have yet to fully optimise the code, for instance we currently do a dense solve for $\mathbf{P} = \mathbf{Y}^{-1}$ which is a bottleneck (this could be heavily optimised or possibly avoided completely). Also, once a landmark has been created and passed the map management test, it always remains in the estimate; however, sliding window estimation is constant time if you choose to marginalise out landmarks i.e. maintain a constant state size. The next chapter introduces a hybrid representation of the map, which allows the SLAMIDE system to update in real-time.

Real-world experiment: We will now show how the system performs on real-world RADAR data taken using a millimetre wavelength RADAR with an 800m range, mounted on a boat moving around in Portsmouth harbour. Figure 4.6 shows an aerial view of the area, the salient points of interest are: several pier structures and a large patch of moored boats.

Figure 4.7 shows the raw RADAR data after automatic gain control and clutter removal, it should be pointed out that there is still a significant clutter density. The RADAR sensor is continually rotating and sends out directional pulses of microwave

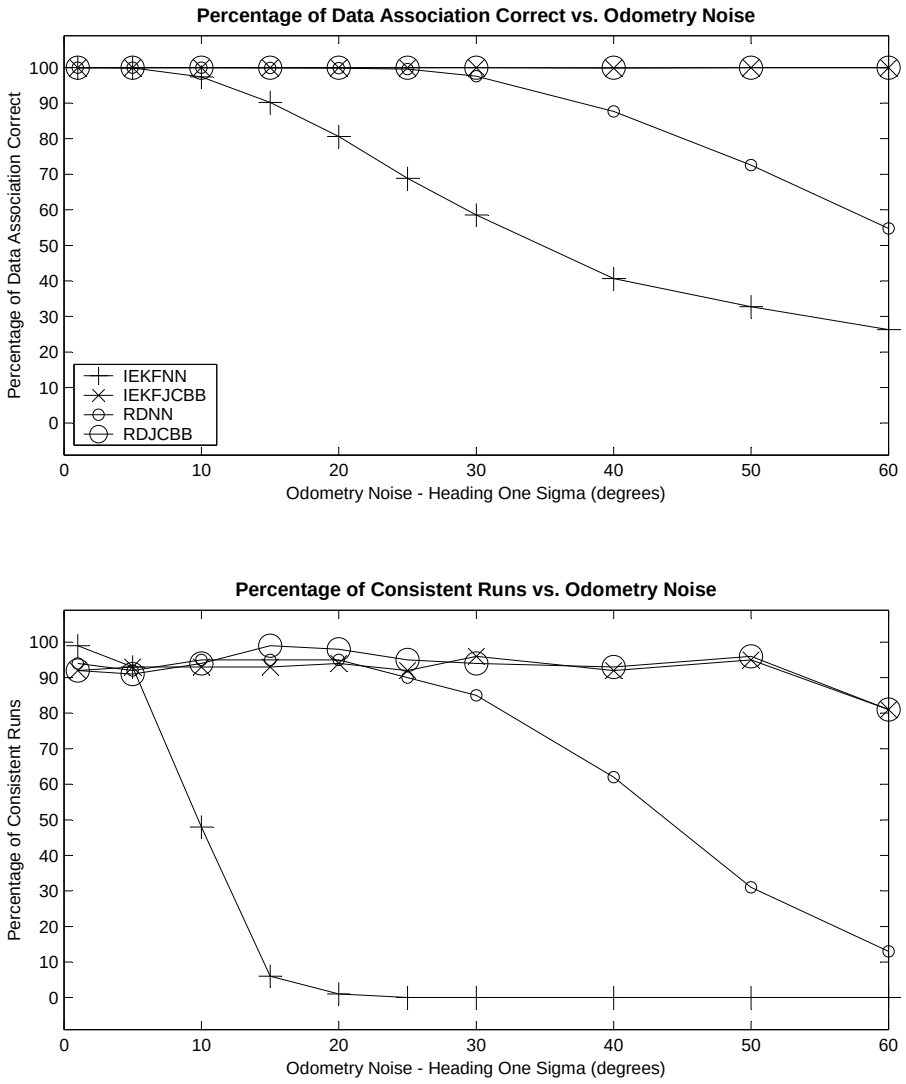


Figure 4.3: Comparison for increasing odometry (rate-of-turn) noise.

radiation (azimuths), objects in the environment may reflect some of this energy and by accurately measuring the time-of-flight it is possible to estimate the range of the object. Typically the RADAR used in these experiments exhibits five to ten thousand individual range bearing measurements per scan (after pre-processing), rotating at 0.67Hz this generates approximately five thousand range bearing measurements per second. This large number of incoming measurements is first clustered using mean shift into small clusters and then the cluster centres are fed into the SLAMIDE system as measurements.

Figure 4.8 shows an example of the output of the SLAMIDE algorithm: the light blue

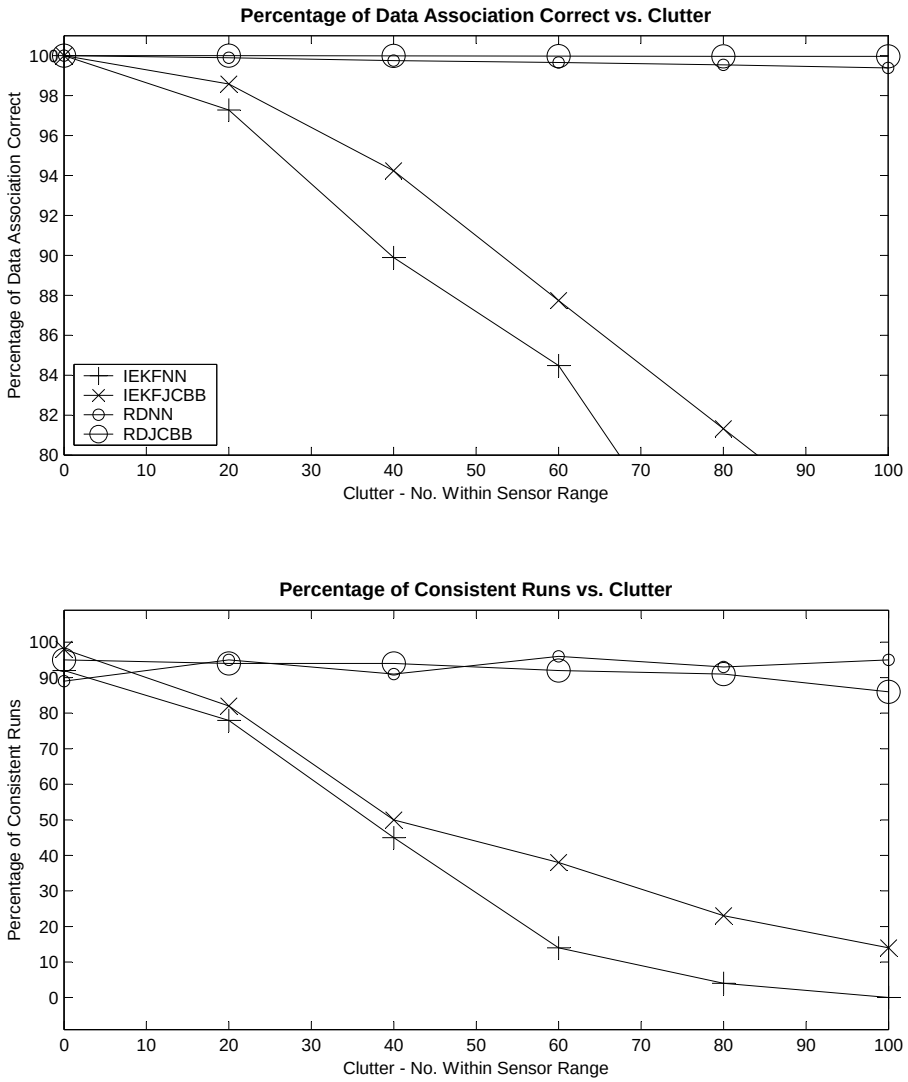


Figure 4.4: Comparison for increasing clutter.

ellipses represent the land mark covariances; the faint blue ellipses are landmarks that are classified as clutter (but not yet deleted from the estimate); the green line is the vehicle's trajectory; the yellow lines are the trajectories of dynamic objects and the red dots are landmarks that are no longer observed in the temporal sliding window. Figure 4.9 shows the output with the clutter hidden, the SLAMIDE algorithm does an excellent job of successfully removing clutter before it gets marginalised into the estimate. Finally, Figure 4.10 shows the output from the SLAMIDE system overlaid on the original aerial view, illustrating that the system does an excellent job of estimating the egomotion of the vessel using raw RADAR alone (no GPS or compass); the system closely matches the rigid pier structure protruding into the river.

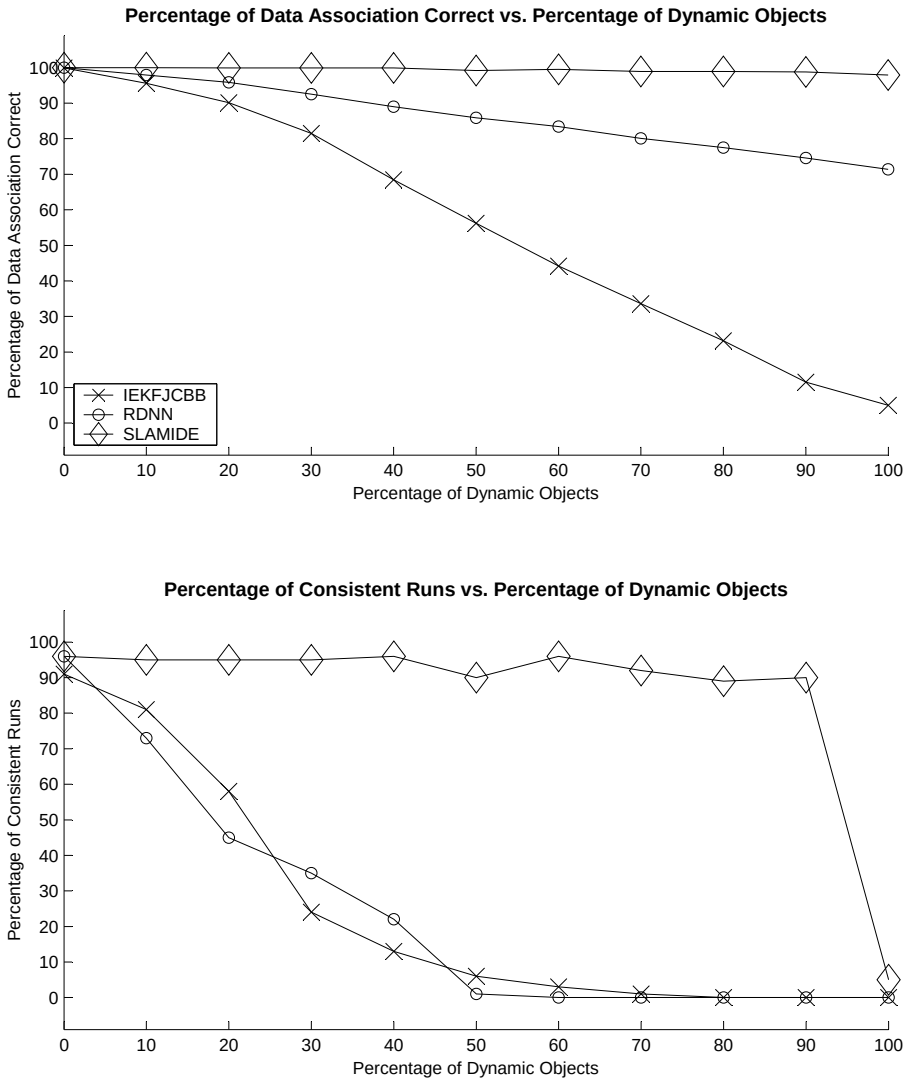


Figure 4.5: Comparison for an increasing percentage of dynamic objects.

4.5 Conclusions

This chapter has described a method that combines sliding window optimisation and least-squares together with generalised expectation maximisation to do reversible model-selection and data-association. This enables dynamic objects to be included directly in the SLAM estimate, as opposed to other techniques which typically detect dynamic objects and then either treat them as outliers [Wolf & Sukhatme, 2004; Hähnel *et al.*, 2003] or track them separately [Wang *et al.*, 2003]. Simulation results show that: (i) our SLAMIDE algorithm significantly outperforms other methods which treat dynamic objects as clutter; (ii) our method for computing reversible

Figure 4.6: Aerial view of Portsmouth harbour.

data-association remains consistent when other data-association methods fail and (iii) our reversible data-association provides excellent performance when clutter is present. Aside from simulation we have also shown our algorithms running on real RADAR data with very good results. We can tell using satellite imagery as ground truth that the system is able to accurately estimate the egomotion. The system correctly classifies the dynamic objects and is able to fuse intermittent measurements over longer periods of time.

In summary, this chapter has described a method for including dynamic objects directly into the SLAM estimate as opposed to treating them as outliers. This has benefits for navigation and path planning; interestingly it also helps with localisation in highly dynamic environments, especially during short periods of time when stationary landmarks are not observed. From a practical perspective there are three problems with the method proposed in this chapter:

1. The land mass is represented using point features. This is achieved by first clustering the incoming raw RADAR data into small clusters and then using the centres of these clusters as the measurements in the SLAMIDE system.

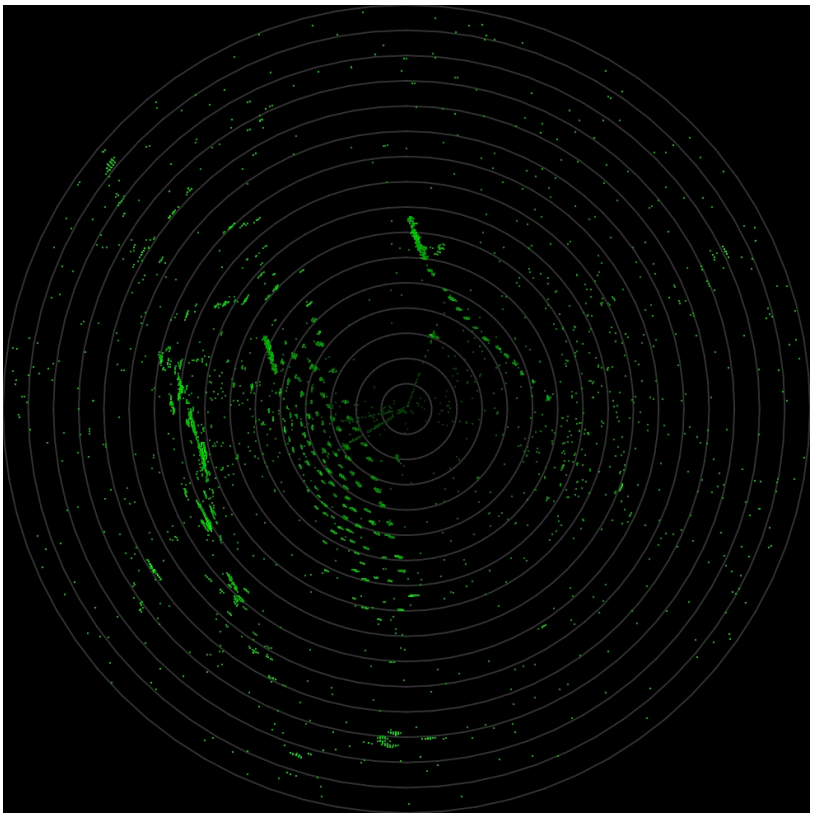


Figure 4.7: Raw RADAR data after pre-processing.

The problem is that large RADAR responses corresponding to land mass will be broken into many small clusters and that these clusters are not guaranteed to fall at the same locations from one scan to the next. This has two negative effects: firstly, the non repeatable cluster centres will inevitably reduce the quality of the estimate and secondly, large landmasses become expensive to represent in the SLAMIDE map because they have to be represented using many small point features.

2. Dynamic object trajectories are expensive in the SLAMIDE estimation because each dynamic object has a fully correlated pose and velocity estimate for each time-step in the temporal sliding window. This is the direct consequence of including dynamic objects directly in the SLAM estimate and proves computationally expensive.

3. The rotating RADAR sensor acquires data continuously but the algorithm only represents the vehicle's pose at discrete time intervals. This presents a problem for the generative model in Figure 4.2, which assumes that measurements are acquired at exactly the same discrete time intervals as the pose estimates. This

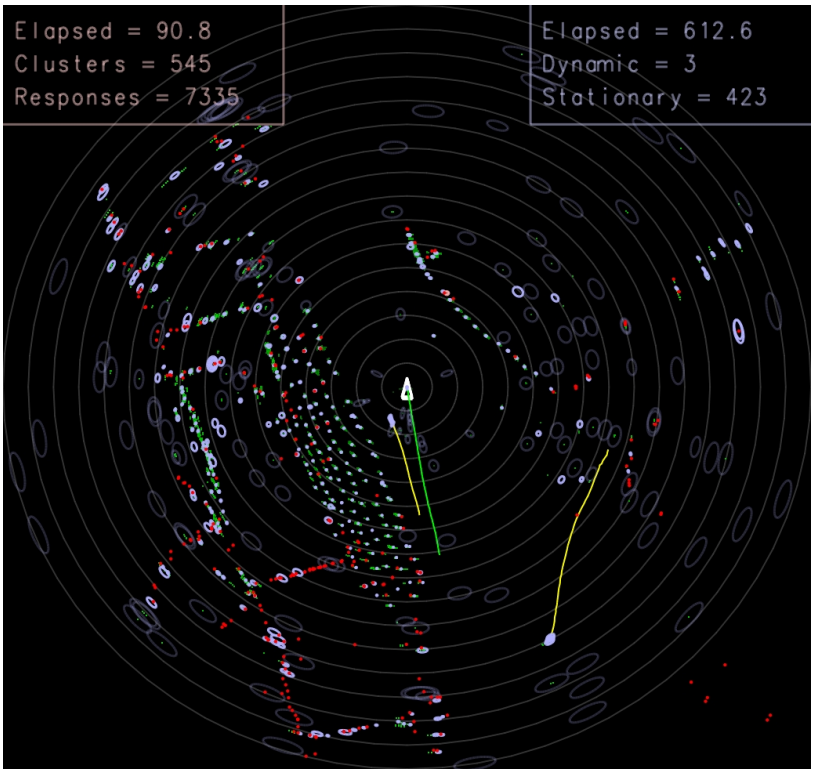


Figure 4.8: The output from SLAMIDE (with clutter displayed).

is handled by ignoring the egomotion of the RADAR sensor during a single scan acquisition. A better way of dealing with this would be to extrapolate the measurements to the nearest discrete time-step using the current estimate for the egomotion.

The next chapter deals with these issues by having a hybrid map representation that uses the point features described in this chapter along with an occupancy grid to efficiently represent landmasses, and cubic splines to represent dynamic object trajectories and the egomotion of the vehicle.

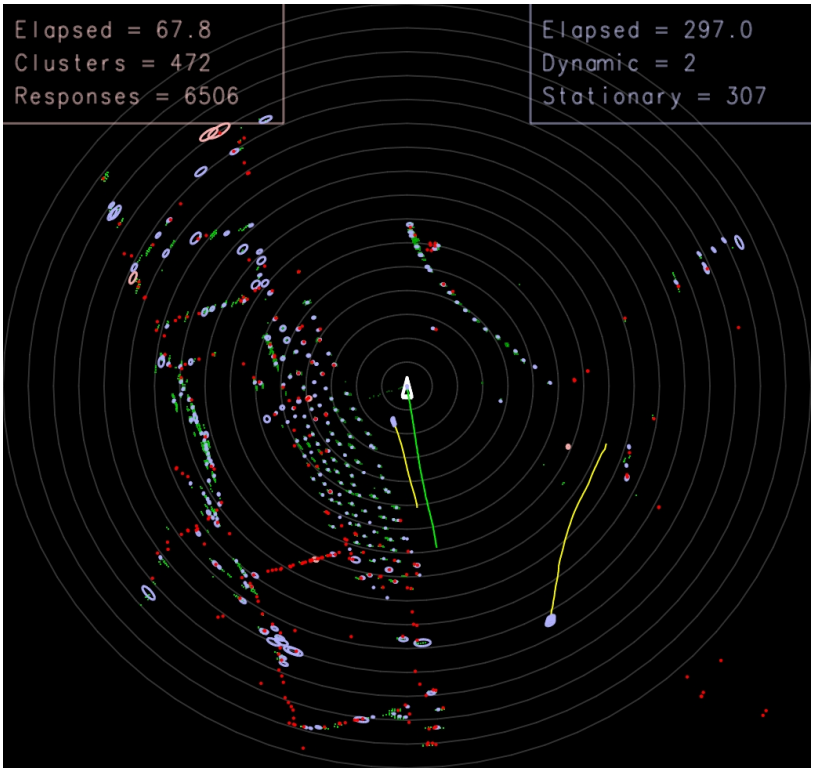


Figure 4.9: The output from SLAMIDE (with clutter hidden).

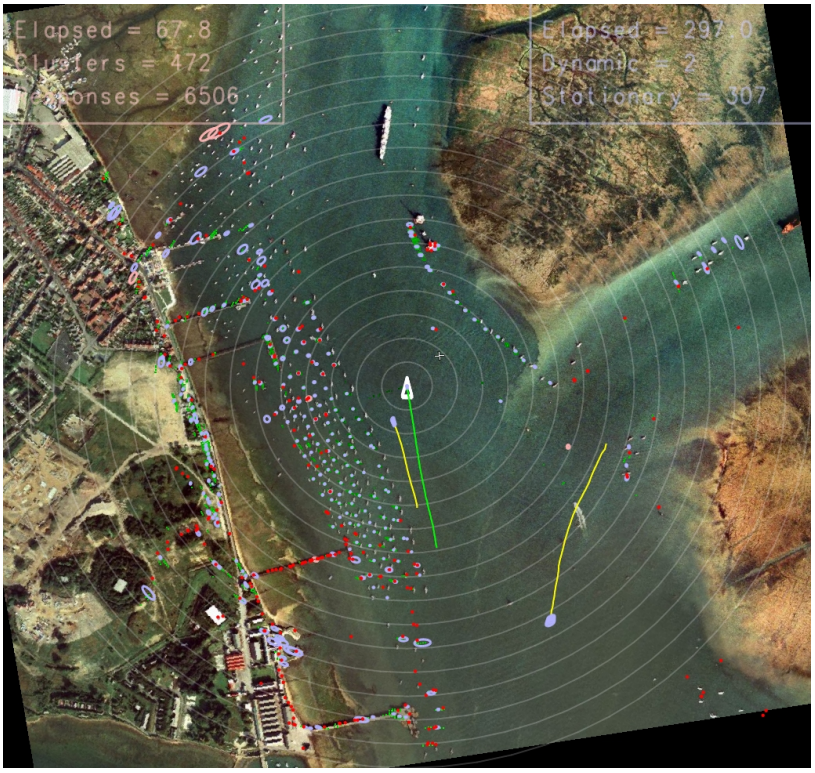


Figure 4.10: The output from SLAMIDE overlaid on the satellite ground truth.

Chapter 5

Hybrid Mapping

5.1 Introduction

The previous chapter presented our approach for doing SLAMIDE, which extended the augmented state representation to include dynamic objects. We showed how the basic algorithm could be applied to real marine RADAR data by making some rather simplistic assumptions i.e. the world can be represented using point features and the egomotion during RADAR acquisition can be ignored. These sort of assumptions are often applied to theoretical frameworks to get them working on real data. This chapter deals with practical aspects and careful implementation details required to produce a system that works reliably in the 'real-world'. This is non-trivial and requires significant amounts of domain knowledge, good underlying representations, carefully selected heuristics and non-trivial tuning. Our 'real-world' scenario is the marine environment observed from a small boat using RADAR. This challenging environment contains landmasses, stationary objects and dynamic objects, and it is desirable to model them all within a single framework. To add to the difficulty, the marine RADAR sensor is prone to poor angular resolution, reflections, interference and clutter. Following our approach from the previous chapter, we tackle these difficulties using sliding window estimation and reversible decision making, and we add hybrid mapping. The temporal sliding window provides a fixed period of time during which the estimated landmasses, stationary and dynamic objects, and the egomotion trajectory can be refined, as well as allowing reversible data-association and model-selection. This is crucial to the success of the system because it is only possible to make the correct decision about the true origin of a measurement (clutter, small

stationary object, landmass or dynamic object), when given enough time to observe temporal characteristics.

The first shortcoming of the method presented in the previous chapter is that point features can not directly represent large objects e.g. landmasses. This is tackled by using a heuristic clustering method to represent these large objects as a mixture of point features. The problem with this approach is that the clustering method will not produce the same cluster centres for different scans, leading to poor data-association and therefore a reduction in overall accuracy. We address this problem by using occupancy grids [Moravec & Elfes, 1985; Elfes, 1989; Burgard *et al.*, 1999] (see Section 3.4). This allows us to deal with objects of arbitrary size and shape probabilistically, by breaking them into small grid cells and then computing the posterior probability that a grid cell is occupied. The second problem is that a pose is dropped at every time-step for all dynamic objects, resulting in the system quickly falling below real-time performance for any significant number of dynamic objects. We tackle this problem by using a cubic spline representation for the trajectories of dynamic objects and the egomotion. This allows the trajectories to be automatically compressed based on their kinematics. In other words, if a dynamic object moves in a straight line for five minutes, the system can represent this using the equivalent of only two poses, one at the beginning and one at the end of the trajectory. In practice the kinematics are rarely that simple and so the system finds the appropriate compromise, which typically results in around a 70-80% reduction in the number of states required in the estimation process. The third problem with the methods in the previous chapter, which is also common to the majority of work in this research area, is that the sensor is treated as a synchronous snapshot of the world, whereas in reality the sensor data is actually acquired whilst the vehicle is moving. This results in errors in the estimate, which is seen in the results section as shadowing in the occupancy grid (caused by measurements falling into the wrong grid cell). Our spline representation allows us to compensate for this by re-rendering the sensor data to account for the egomotion undergone during the sensor acquisition period. The temporal sliding window is used to continually refine the egomotion trajectory and the occupancy grid (similar to [Thrun *et al.*, 1998]) using generalised expectation maximisation [Neal & Hinton, 1998; Dempster *et al.*, 1977]. This improves the estimate of the egomotion, the occupancy grid and the dynamic objects as well as providing the necessary time required to get data-association and model-selection correct.

The notion of hybrid mapping is becoming increasingly popular, with [Pandey *et al.*,

2007] incorporating features within an occupancy grid framework and [Nieto *et al.*, 2004] breaking the occupancy grid into triangular patches and using feature based methods to estimate their positions. Our method of using cubic splines to represent trajectories within a SLAM system is the first of its kind and provides an elegant and compact way of representing trajectories in a continuous manner. The closest related work is [Pedraza *et al.*, 2007], which uses Bezier splines to represent stationary objects i.e. walls and corridors. In contrast, our method uses splines to represent the trajectories of dynamic objects and hence the spline parameter represents time. This has three major advantages: (i) the number of parameters required for the spline is less than having a pose at each time-step; (ii) the continuous nature of the spline makes it trivial to add measurements to the system at arbitrary times, making it easy to use asynchronous measurements from sensors running at different frequencies and (iii) it is now possible to compute a position/velocity at any point in time along a trajectory, which allows sensor scans to be re-rendered at a sub-scan resolution to compensate for the egomotion during the period the scan was acquired.

This chapter begins in Section 5.2 by introducing the notation and showing how to use a hybrid map with SLAM; Section 5.3 explains how to use cubic splines within the framework; Section 5.4 shows the results of using the system on real RADAR data and analyses the effect of the cubic spline representation and finally Section 5.5 concludes.

5.2 Hybrid SLAM in Dynamic Environments

We will now introduce our method for Hybrid SLAM in Dynamic Environments (HSLAMIDE). The most significant changes from traditional SLAM are: (i) we use a hybrid representation using occupancy grids, cubic splines and point features; (ii) the map becomes time dependent and (iii) model-selection parameters are introduced. For simplicity we will begin by explaining the system without using cubic splines and then in Section 5.3 we will demonstrate how to retrofit splines to the system. Below is a list of the notation we will be using:

- τ: The beginning of the temporal sliding window.
- T: The end of the temporal sliding window.
- $\mathbf{x}_t$: The state vector at time t describing the vehicle's pose (location and orien-

tation $[x, y, \theta]$).

- $\mathbf{u}_t$: The control vector (odometry $[\dot{x_v}, \dot{y_v}, \dot{\theta_v}]$ in vehicle coordinates where $\dot{x_v}$ is in the direction the vehicle is pointing) that was applied to vehicle at time $t-1$ to take it to time t.
- $\mathbf{S}_t = \{\mathbf{z}_t, \mathbf{r}_t\}$: A complete RADAR scan obtained at time t. This is decomposed into range-bearing measurements $\mathbf{z}_t$ and a residual RADAR scan $\mathbf{r}_t$.
- $\mathbf{z}_t$: Range-bearing measurements extracted from the RADAR scan $\mathbf{S}_t$ based on a constraint on the permissible object size.
- $\mathbf{r}_t$: The residual RADAR scan after the range-bearing measurements have been extracted.
- $\mathbf{m}^k$: State vector describing the location of object k.
- $\mathbf{O}$: The occupancy grid representing landmasses.
- $\mathbf{M}_t = \{\mathbf{m}_t^1, \ldots, \mathbf{m}_t^k\}$: The set of all objects at time t.
- $\mathbf{X}_{\tau:T} = \{\mathbf{x}_\tau, \ldots, \mathbf{x}_T\}$: The set of vehicle poses.
- $\mathbf{U}_{\tau:T} = \{\mathbf{u}_\tau, \ldots, \mathbf{u}_T\}$: The set of odometry.
- $\mathbf{Z}_{\tau:T} = \{\mathbf{S}_\tau, \ldots, \mathbf{S}_T\}$: The set of all measurements i.e. the residual RADAR scans plus the extracted range-bearing measurements.
- $\mathbf{M}_{\tau:T} = \{\mathbf{M}_\tau, \ldots, \mathbf{M}_T\}$: The map consisting of stationary and dynamic objects.
- $\mathbf{D}_{\tau:T} = \{\mathbf{d}_\tau, \ldots, \mathbf{d}_t\}$: The data-association.
- $\mathbf{V} = \{\mathbf{v}^1, \ldots, \mathbf{v}^k\}$: The model-selection parameters.

Figure 5.1 is a graphical model that shows our formulation of the HSLAMIDE problem. The joint distribution corresponding to Figure 5.1 is:

$$P(\mathbf{X}_{\tau:T}, \mathbf{O}, \mathbf{M}_{\tau:T}, \mathbf{D}_{\tau:T}, \mathbf{V}, \mathbf{U}_{\tau:T}, \mathbf{Z}_{\tau:T}) =$$
$$P(\mathbf{x}_\tau)P(\mathbf{O})P(\mathbf{M}_\tau)P(\mathbf{D}_{\tau:T})P(\mathbf{V})\times$$
$$\prod_{t=\tau+1}^{T}\Big\{P(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M}_t, \mathbf{d}_t)P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)\times$$
$$P(\mathbf{M}_t|\mathbf{M}_{t-1}, \mathbf{V})P(\mathbf{r}_t|\mathbf{x}_t, \mathbf{O})\Big\}, \quad (5.1)$$

where:

- $P(\mathbf{x}_\tau)$ is the Gaussian prior on vehicle state at the beginning of the temporal sliding window, which has a mean $\tilde{\mathbf{x}}_\tau$ and covariance $\mathbf{P}_\tau^0$.
- $P(\mathbf{O})$ is the prior on the occupancy grid at the beginning of the sliding window and is taken to be 0.4 for each grid cell.
- $P(\mathbf{M}_\tau)$ is the Gaussian prior on the map and is treated independently for each object i.e. $\prod_1^K P(\mathbf{m}_\tau^k)$.
- $P(\mathbf{D}_{\tau:T})$ is the prior on the data-association and is taken to be the uninformative uniform distribution.
- $P(\mathbf{V})$ is the prior on the model-selection parameters and assumes that new objects are dynamic (refer to Step 3 of the algorithm for details).
- $P(\mathbf{z}_t|\mathbf{x}_t, \mathbf{M}_t, \mathbf{d}_t)$ is the Gaussian measurement model for objects in the map i.e. the probability of the measurement $\mathbf{z}_t$ given the vehicle pose $\mathbf{x}_t$, the map $\mathbf{M}$ and the data-association $\mathbf{d}_t$.
- $P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t)$ is the Gaussian motion model i.e. the probability of the new vehicle pose $\mathbf{x}_t$ given the last vehicle pose $\mathbf{x}_{t-1}$ and the odometry $\mathbf{u}_t$.
- $P(\mathbf{M}_t|\mathbf{M}_{t-1}, \mathbf{V})$ is the Gaussian motion model for the map given the current estimate of the model-selection parameters. We use a constant position model with zero noise for stationary objects and three constant velocity models with noise in $\dot{x}$ and $\dot{y}$ for dynamic objects.
- $P(\mathbf{r}_t|\mathbf{x}_t, \mathbf{O})$ is the Gaussian measurement model for a single residual RADAR scan i.e. the probability of the RADAR scan $\mathbf{r}_t$ given the vehicle pose $\mathbf{x}_t$ and the occupancy grid $\mathbf{O}$.

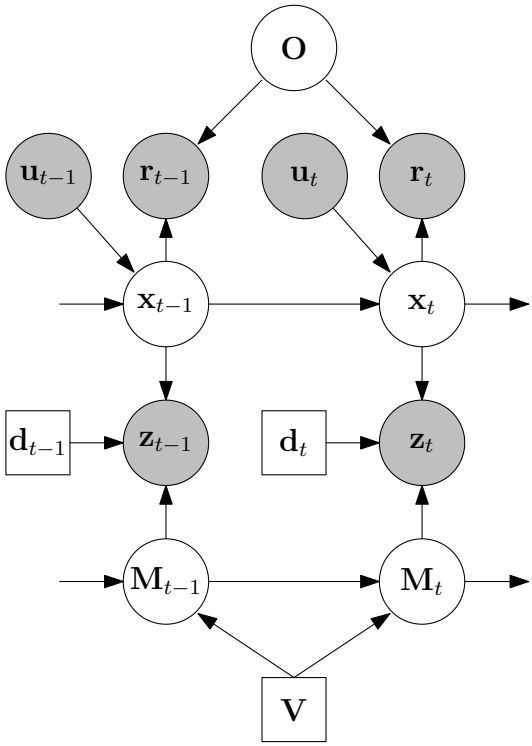


Figure 5.1: A graphical model representing HSLAMIDE.

Solving (5.1) with a single optimisation is intractable and so we propose the following steps based on generalised expectation maximisation [Neal & Hinton, 1998] to solve the optimisation in real-time (see Chapter 4 for details):

1. $\mathbf{D}' = \arg\max_{\mathbf{D}} P(\mathbf{X}, \mathbf{O}, \mathbf{M}, \mathbf{D}, \mathbf{V}, \mathbf{U}, \mathbf{Z})$
2. $\{\mathbf{X}', \mathbf{M}'\} = \arg\max_{\{\mathbf{X},\mathbf{M}\}} P(\mathbf{X}, \mathbf{O}, \mathbf{M}, \mathbf{D}', \mathbf{V}, \mathbf{U}, \mathbf{Z})$
3. $\mathbf{V}' = \arg\max_{\mathbf{V}} P(\mathbf{X}', \mathbf{O}, \mathbf{M}', \mathbf{D}', \mathbf{V}, \mathbf{U}, \mathbf{Z})$
4. $\mathbf{O}' = \arg\max_{\mathbf{O}} P(\mathbf{X}', \mathbf{O}, \mathbf{M}', \mathbf{D}', \mathbf{V}', \mathbf{U}, \mathbf{Z})$

Note:- The subscript $_{\tau:T}$ has been dropped from all terms to save space and the $'$ notation indicates the new estimate. We will now explain in detail each of the steps.

Step 1: performs the data-association using a probabilistic data-association filter [Bar-Shalom & Fortmann, 1988] with an initial Mahalanobis gate of four and a uniform distribution modeling the outlier process. This method allows uncertainty in data-association to be modelled and adds robustness to outliers.

Step 2: is a least-squares optimisation for the vehicle trajectory and the objects in the map. Taking the logarithm of (5.1) we can write:

$$\{\mathbf{X}^{'}, \mathbf{M}^{'}\} = \underset{\{\mathbf{X},\mathbf{M}\}}{\arg\min}\Bigg\{ \|\tilde{\mathbf{x}}_\tau - \mathbf{x}_\tau\|^2_{\mathbf{P}^0_\tau} + \sum_{k=1}^{K} \left\|\tilde{\mathbf{m}}^k_\tau - \mathbf{m}^k_\tau\right\|^2_{\mathbf{P}^k_\tau} + \sum_{t=\tau+1}^{T} \Bigg\{ \|f_x(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t\|^2_{\mathbf{Q}^0} + \|g(\mathbf{r}_t, \mathbf{O}) - \mathbf{x}_t\|^2_{\mathbf{B}} + \sum_{k=1}^{K} \Big(\left\|f_m(\mathbf{m}^k_{t-1}, v_k) - \mathbf{m}^k_t\right\|^2_{\mathbf{Q}^k_{v_k}} + \left\|h(\mathbf{x}_t, \mathbf{m}^{d^k_t}_t) - \mathbf{z}_t\right\|^2_{\mathbf{R}} \Big) \Bigg\} \Bigg\}, \quad (5.2)$$

where $\|\tilde{\mathbf{x}}_\tau - \mathbf{x}_\tau\|^2_{\mathbf{P}^0_\tau}$ is the Gaussian prior on the vehicle's position/velocity at the beginning of the temporal sliding window. $\left\|\tilde{\mathbf{m}}^k_\tau - \mathbf{m}^k_\tau\right\|^2_{\mathbf{P}^k_\tau}$ is the Gaussian prior on the object's position/velocity at the beginning of the temporal sliding window. $\|f_x(\mathbf{x}_{t-1}, \mathbf{u}_t) - \mathbf{x}_t\|^2_{\mathbf{Q}^0}$ is the motion model taken to be constant velocity with covariance $\mathbf{Q}^0$. $\|g(\mathbf{r}_t, \mathbf{O}) - \mathbf{x}_t\|^2_{\mathbf{B}}$ is the registration between a single RADAR scan $\mathbf{r}_t$ and the occupancy grid $\mathbf{O}$, giving an estimate of the vehicle's pose (location and orientation $[x, y, \theta]$) with covariance $\mathbf{B}$. This registration is computed using our pixel-wise posterior based registration technique that will be described in Chapter 6. $\left\|f_m(\mathbf{m}^k_{t-1}) - \mathbf{m}^k_t\right\|^2_{\mathbf{Q}^k_{v_k}}$ is the motion model of an object, which can be one of four motion models: stationary with zero uncertainty or constant velocity with three different levels of covariance $\mathbf{Q}^k_{v_k}$. Finally, $\left\|h(\mathbf{x}_t, \mathbf{m}^{d^k_t}_t) - \mathbf{z}_t\right\|^2_{\mathbf{R}}$ is a range-bearing measurement model with covariance $\mathbf{R}$ for the measurement $\mathbf{z}_t$.

Step 3: computes the model-selection parameters $\mathbf{V}^{'}$ and is computed using the following discrete Bayesian update:

$$P(v_k|\mathbf{m}^k_t) = \frac{P(\mathbf{m}^k_t|v_k)P(v_k|\mathbf{m}^k_{t-1})}{P(\mathbf{m}^k_t)}. \quad (5.3)$$

$P(v_k|\mathbf{m}^k_{t-1})$ is initially set to $[0.1, 0.3, 0.3, 0.3]$ where the first element corresponds to the stationary model and then the next three consecutive elements correspond to increasing amounts of motion model noise. The term $P(\mathbf{m}^k_t|v_k)$ is the likelihood for a given model and is a Gaussian on velocity with standard deviations of $[1, 5, 10, 20]$ knots for the four models respectively. We then take the MAP estimate for use in the motion model. It is worth noting that once the stationary model is selected there is no way to go back to dynamic, since the stationary motion model is a hard constant

position and hence any subsequent velocities are zero (in practice this is dealt with by adding a new dynamic landmark).

Step 4: re-renders the occupancy grid based on the updated vehicle trajectory $\mathbf{X}'$ and the RADAR scans $\{\mathbf{r}_\tau, \ldots, \mathbf{r}_T\}$. This step is implemented using the Graphics Processing Unit (GPU) and makes the standard assumption of independence between grid cells (3.1), see Section 3.4 and [Thrun *et al.*, 2005] for details. The posterior for each grid cell is computed using floating point textures and fragment shaders on the GPU. The benefit of this is that several minutes worth of RADAR data can be re-rendered within a few milliseconds on standard hardware. Our underlying cubic spline representation (see Section 5.3) allows us to re-render at a sub-scan resolution i.e. we can compensate for the egomotion of the vehicle during the time taken to acquire a RADAR scan.

Multi-threading: In practice this whole process is implemented in four separate threads: thread one computes Steps 1,2 and 3; thread two computes Step 4; thread three computes the registrations required for Step 2 and thread four performs pre-processing and clustering on the RADAR scans.

New objects: These are added as part of the data-association step. A new object must have a Mahalanobis distance of at least sixteen from every existing object. The reasoning behind this is that we want to be absolutely sure that the measurement generating a new object was not generated from an object in the system. Once a new object is detected, it is then added using the predicted object location (given the vehicle pose and the measurement) and the corresponding uncertainty.

Object deletion: We have two criteria for deleting objects: (i) if no measurements are associated to it within the temporal sliding window and (ii) if the measurement density is less than 30% during the first 10 seconds of an object's life.

Object merging: Because we use a probabilistic data-association filter it is possible that two initially separate objects can converge to the same trajectory by sharing measurements. We deal with this problem by measuring the sum-of-squared differences between overlapping trajectories and if sufficiently small we merge the two objects.

5.3 Cubic Splines as a Continuous Trajectory Representation

We will now introduce the novel idea of using cubic splines to represent trajectories. The key concept is to represent a trajectory $[\mathbf{x}_0, \ldots, \mathbf{x}_t]$ as a set of cubic spline sections, which requires a significantly smaller parameter set compared with the requirement for a full representation. For instance, if a 2D vehicle is moving with a constant rate of change of acceleration for 2 seconds and its fastest sensor runs at 100Hz (e.g. an inertial sensor), then those two hundred poses (200 poses $\times$ 3 parameters/pose = 600 parameters) can now be represented by 12 spline parameters i.e. 2% of the original size. This huge compression then allows us to solve much larger temporal sliding window lengths. The second major benefit is that the trajectory now has a continuous representation rather than discrete time-steps, which makes it easy to deal with asynchronous measurements/constraints within the system. Figure 5.2 illustrates the difference between a traditional approach and our method using splines. The top graphical model represents a traditional approach, where for each incoming measurement $\mathbf{Z} = [\mathbf{z}_1, \ldots, \mathbf{z}_5]$ (e.g. a RADAR, SONAR or laser scan) there is a corresponding pose in the state vector $\mathbf{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_5]$. In contrast, our method uses a cubic spline section described by the two knots $[\mathbf{Y}_0, \mathbf{Y}_1]$ to represent the poses. This has two consequences: (i) each measurement constraint in the original graphical model is now projected into two constraints, one for each knot and (ii) the motion model constraints are now represented by a single constraint between the knots. We will now explain how the top graphical model can be solved using linear algebra and then elaborate on how to introduce cubic splines, highlighting the required modifications.

The Traditional Solution

The joint distribution for the top graphical model is:

$$P(\mathbf{X}, \mathbf{Z}) = P(\mathbf{x}_0) \prod_{t=1}^{5} P(\mathbf{z}_t|\mathbf{x}_t) P(\mathbf{x}_t|\mathbf{x}_{t-1}), \tag{5.4}$$

where $P(\mathbf{x}_0)$ is the prior, $P(\mathbf{z}_t|\mathbf{x}_t)$ is a Gaussian measurement model and $P(\mathbf{x}_t|\mathbf{x}_{t-1})$ is a Gaussian motion model. Let us now take the logarithm of (5.4) to obtain a

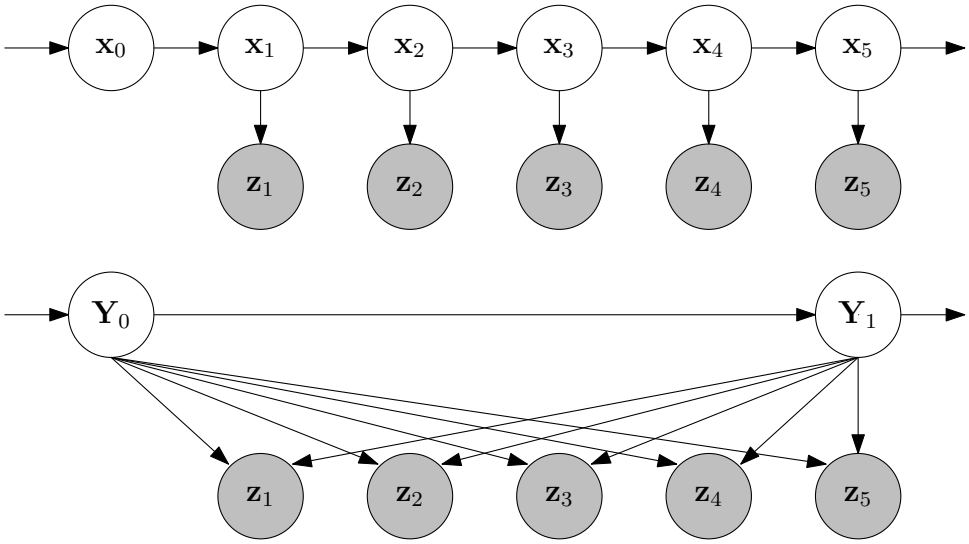


Figure 5.2: Graphical models: (top) a traditional formulation and (bottom) our method using splines.

non-linear least-squares problem:

$$\mathbf{X} = \arg\min_{\mathbf{X}} \left\{ \|\tilde{\mathbf{x}}_0 - \mathbf{x}_0\|^2_{\mathbf{P}} + \sum_{t=\tau+1}^{T} \left(\|f(\mathbf{x}_{t-1}) - \mathbf{x}_t\|^2_{\mathbf{Q}} + \|h(\mathbf{x}_t) - \mathbf{z}_t\|^2_{\mathbf{R}} \right) \right\}, \tag{5.5}$$

where $f(.)$ is the motion model with covariance $\mathbf{Q}$ and $h(.)$ is the measurement model with covariance $\mathbf{R}$. This minimisation can be solved by linearising the non-linear terms and re-writing as a matrix equation:

$$\begin{aligned} \hat{\mathbf{X}} = \arg\min_{\mathbf{X}} \Big\{ & \|\boldsymbol{\delta}\mathbf{x}_0 - \{\mathbf{x}_0 - \tilde{\mathbf{x}}_0\}\|^2_{\mathbf{P}} + \\ & \sum_{t=1}^{T} \big(\|\{\mathbf{F}_{t-1}\boldsymbol{\delta}\mathbf{x}_{t-1} + \boldsymbol{\delta}\mathbf{x}_t\} - \{\mathbf{x}_t - f(\mathbf{x}_{t-1})\}\|^2_{\mathbf{Q}} + \\ & \|\{\mathbf{H}_t\boldsymbol{\delta}\mathbf{x}_t\} - \{\mathbf{z}_t - h(\mathbf{x}_t)\}\|^2_{\mathbf{R}} \big) \Big\}, \end{aligned} \tag{5.6}$$

where $\mathbf{F}_{t-1}$ is the Jacobian of $f(.)$ with respect to $\mathbf{x}_{t-1}$ and $\mathbf{H}_t$ is the Jacobian $h(.)$ with respect to $\mathbf{x}_t$. Equation (5.6) can be factorised and written as a system of linear equations ($\mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{A}\boldsymbol{\delta} = \mathbf{A}^T\boldsymbol{\Sigma}^{-1}\mathbf{b}$); for a detailed look at this process refer to Section 3.7.

Using Cubic Splines

Each cubic spline is constructed with N piecewise third-order polynomials passing through control points (knots) $[\mathbf{Y}_0, \ldots, \mathbf{Y}_N]$. These knots are parametrised by their position $\mathbf{y}_n = \{x_n, y_n, \theta_n\}$ and velocity $\dot{\mathbf{y}}_n = \{\dot{x}_n, \dot{y}_n, \dot{\theta}_n\}$. If p is a parameter in the range $[0 \ldots 1]$ and n references one of the N spline sections then a single spline section is defined as:

$$\mathbf{ss}_n(p) = \mathbf{a}_n + \mathbf{b}_n p + \mathbf{c}_n p^2 + \mathbf{d}_n p^3$$

where

$$\begin{aligned}
\mathbf{a}_n &= \mathbf{y}_n \\
\mathbf{b}_n &= \dot{\mathbf{y}}_n \\
\mathbf{c}_n &= 3(\mathbf{y}_{n+1} - \mathbf{y}_n) - 2\dot{\mathbf{y}}_n - \dot{\mathbf{y}}_{n+1} \\
\mathbf{d}_n &= 2(\mathbf{y}_n - \mathbf{y}_{n+1}) + \dot{\mathbf{y}}_n + \dot{\mathbf{y}}_{n+1}.
\end{aligned}$$

Let us now consider the vehicle's trajectory as a spline. Given the spline parameters $\mathbf{\Phi} = \{\mathbf{Y}_0, \ldots, \mathbf{Y}_N\}$ and a time t we can write a function that returns the vehicle's state:

$$\begin{aligned}
\mathbf{s}(t, \mathbf{\Phi}) = \begin{bmatrix} x_n(p) \\ y_n(p) \\ \theta_n(p) \end{bmatrix} &= \begin{bmatrix} a_n^x + b_n^x p + c_n^x p^2 + d_n^x p^3 \\ a_n^y + b_n^y p + c_n^y p^2 + d_n^y p^3 \\ a_n^\theta + b_n^\theta p + c_n^\theta p^2 + d_n^\theta p^3 \end{bmatrix} \\
n &= \left\lfloor \frac{t}{\lambda} \right\rfloor \\
p &= \frac{t}{\lambda} - n,
\end{aligned} \tag{5.7}$$

where λ is the number of time-steps per spline section, which for simplicity is assumed constant, and the function $\lfloor . \rfloor$ computes the greatest integer less than the argument. In practice we allow λ to change and so solving for n and p turns into a binary search. We also require the Jacobian of $\mathbf{s}(t, \mathbf{\Phi})$ with respect to $\mathbf{\Phi}$:

$$\mathbf{S}_t = \frac{\partial s(t, \mathbf{\Phi})}{\partial \mathbf{\Phi}} =$$
$$\begin{bmatrix} k & 0 & 0 & l & 0 & 0 & m & 0 & 0 & n & 0 & 0 \\ 0 & k & 0 & 0 & l & 0 & 0 & m & 0 & 0 & n & 0 \\ 0 & 0 & k & 0 & 0 & l & 0 & 0 & m & 0 & 0 & n \end{bmatrix}$$

where

$$\begin{aligned} k &= 1 - 3p^2 + 2p^3 \\ l &= p - 2p^2 + p^3 \\ m &= 3p^2 - 2p^3 \\ n &= -p^2 + p^3. \end{aligned}$$

The Jacobian $\mathbf{S}_t$ only depends upon the scalar value p and is therefore suitable for a look up table implementation.

To modify (5.5) and (5.6) to a spline representation we make the direct substitution $\mathbf{X} = \mathbf{\Phi}, \mathbf{x}_t = \mathbf{s}(t, \mathbf{\Phi})$ and include $\mathbf{S}_t$ to project any Jacobians with respect to $\mathbf{x}_t$ onto the spline parameter set $\mathbf{\Phi}$, here are the modified versions:

$$\mathbf{\Phi} = \arg\min_{\mathbf{\Phi}} \Big\{ \|\tilde{\mathbf{x}}_0 - \mathbf{s}(0, \mathbf{\Phi})\|^2_{\mathbf{P}} + \sum_{t=\tau+1}^{T} \big(\|f(\mathbf{s}(t-1, \mathbf{\Phi})) - \mathbf{s}(t, \mathbf{\Phi})\|^2_{\mathbf{Q}} + \|h(\mathbf{s}(t, \mathbf{\Phi})) - \mathbf{z}_t\|^2_{\mathbf{R}} \big) \Big\}. \quad (5.8)$$

and

$$\hat{\mathbf{\Phi}} = \arg\min_{\mathbf{\Phi}} \Big\{ \|\mathbf{S}_0 \boldsymbol{\delta}\mathbf{\Phi} - \{\mathbf{s}(0, \mathbf{\Phi}) - \tilde{\mathbf{x}}_0\}\|^2_{\mathbf{P}} + \sum_{t=1}^{T} \big(\|\{\mathbf{H}_t \mathbf{S}_t \boldsymbol{\delta}\mathbf{\Phi}\} - \{\mathbf{z}_t - h(\mathbf{s}(t, \mathbf{\Phi}))\}\|^2_{\mathbf{R}} + \|\{\mathbf{F}_{t-1}\mathbf{S}_{t-1}\boldsymbol{\delta}\mathbf{\Phi} - \mathbf{S}_t\boldsymbol{\delta}\mathbf{\Phi}\} - \{\mathbf{s}(t, \mathbf{\Phi}) - f(\mathbf{s}(t-1, \mathbf{\Phi}))\}\|^2_{\mathbf{Q}} \big) \Big\}. \quad (5.9)$$

Consequences Of The Spline Representation

By parameterising a set of poses with a cubic spline section, we are setting a hard motion model (fixed rate of change of acceleration $\dddot{\mathbf{x}}$) for each spline section i.e. between the knots. This is acceptable as most vehicles in the 'real-world' have significant amounts of inertia and are therefore likely to obey relatively smooth trajectories, at least on some scale. This fact is used in typical Kalman Filter style solutions to give a prior on the likely motion a vehicle might undergo. Our cubic spline representation assumes a fixed rate of change of acceleration along the spline section and is therefore able to deal with constant position, constant velocity, constant acceleration and constant rate of change of acceleration motion models. Therefore our cubic spline method is able to not only capture the dynamics of our marine vessel, but also the large majority of vehicles that are typically modelled in the robotics community.

Let us now consider two cases: (i) what happens along a spline section and (ii) what happens at the knots. The hard motion model is enforced along the spline section implicitly because of the cubic spline representation. At the knots, we fix $\mathbf{x}$ and $\dot{\mathbf{x}}$ to be equal (C1 continuity) but allow $\ddot{\mathbf{x}}$ and $\dddot{\mathbf{x}}$ to change to anything i.e. a uniform probability distribution over $\ddot{\mathbf{x}}$ and $\dddot{\mathbf{x}}$.

Knot Placement

The ideal way to place the knots is at moments in time where $\ddot{\mathbf{x}}$ and/or $\dddot{\mathbf{x}}$ change. We currently deal with this problem by initially over representing the trajectory with a knot for every pose (the traditional solution) and then removing those where $\ddot{\mathbf{x}}$ and $\dddot{\mathbf{x}}$ are sufficiently similar. Therefore, knots only remain where they are required to accurately represent the trajectory and the system will automatically place the correct number of knots based on the current environmental conditions e.g. more in a rough sea than in a calm sea. We explore the effect of this in more detail in the subsequent results section.

5.4 Results

We have tested the system on RADAR data obtained from a 50 minute voyage on a small boat using a Garmin GMR18. Figure 5.3 shows a sample of the raw RADAR data. We use a sliding window length of 90 seconds to achieve the temporal fusion

of the RADAR data. Figures 5.4 and 5.5 show examples of the typical output of the system. The occupancy grid is drawn in the background and overlaid with the stationary and dynamic objects (each object in the system is given a unique identifier). The dynamic objects are represented by cubic splines, which are drawn in black and the covariance (which can be computed for any time along the spline) is drawn at regular intervals in a light grey.

The left-hand column of Figure 5.8 shows the results obtained if the occupancy grid is only fused once at the head of the temporal sliding window, as it would be in a filtering style solution to the problem. There are three clear inadequacies visible in the figure: (i) the occupancy grid exhibits incorrect shadowing effects (caused by RADAR data falling into the wrong grid cell); (ii) the range-bearing measurements extracted from the RADAR sweeps have large amounts of noise and (iii) the system suffers from false-negatives and false-positives. All of these problems stem from a poor estimate of the egomotion. In contrast, the right-hand column of Figure 5.8 shows the results of our proposed method using a 90 second temporal sliding window to continually re-estimate the occupancy grid, egomotion, dynamic objects, model-selection and data-association. The results are that: (i) the occupancy grid is clear and sharp, giving a good representation of the surrounding environment; (ii) the range-bearing measurements have significantly less noise on them, making it easier to obtain better estimates of the trajectories of dynamic objects and (iii) because of the reduced noise in the range-bearing measurements the model-selection is better at correctly classifying the objects.

Figure 5.6 shows how a sampled spline trajectory from the voyage degrades as an increasing number of knots are removed. The figure is generated by placing a knot at every time-step (equivalent to the traditional representation), then progressively removing one knot at a time based on which will have the smallest effect (see Section 5.3) and then computing the RMS error between the traditional trajectory and the one using the spline. The figure shows that 80-90% of the knots can be removed before seeing a significant hit in the quality of the trajectory (we have observed similar results in terms of the quality of the occupancy grid against knot reduction). It is worth noting at this point that errors of 1m are small given the 10m vessel and the RADAR sensor which has a range of 2.8km and a resolution of approximately 9m.

Figure 5.7 shows the distribution of the compression ratio (percentage of knots removed with respect to the full representation) for the spline trajectories, this distribution is generated using samples taken from the 50 minute voyage at 2 second intervals.

The results are that on average the spline trajectories are achieving a compression ratio of 70-90%, which matches well with the sampled trajectory in Figure 5.6. This gives an indication that the automatic knot removal procedure is leaving knots at the correct locations and shows that a significant compression is achieved resulting in a smaller system to solve and hence reducing the required computation.

Figure 5.9 shows the results from an 18 minute voyage. At the beginning of the run the output from the system is manually aligned to an Admiralty chart[1]. This area of coast is protected by a sea-wall, which the RADAR clearly picks up, making the manual alignment to the chart straightforward. After 18 minutes the output from the system remains well aligned with the ground truth. This shows that the HSLAMIDE is doing a good job of estimating the egomotion, even though it is using RADAR as the only sensor. We also have the ability to align the output of the system using a GPS sensor and a compass sensor. With this particular example we are able to localise the vehicle's position more accurately using the HSLAMIDE system and manual alignment, than by using the GPS and compass.

5.5 Conclusions

We have demonstrated how hybrid mapping and sliding window estimation can be used to make a SLAM system that works in complicated dynamic environments using noisy sensors. Our hybrid mapping gives a rich representation of the underlying environment, with an occupancy grid to represent land masses, point features to represent smaller stationary objects and cubic splines to represent the trajectories of dynamic objects. We have also shown how cubic splines can easily be retrofitted to an estimation framework containing dynamic objects and how this achieves state compression, makes it easy to deal with asynchronous measurements from sensors running at different frequencies and allows us to re-render sensor data to compensate for the egomotion during the sensor acquisition period. This hybrid mapping system allows us to do SLAM in dynamic environments that were too difficult for the method presented in the previous chapter.

The key differences of this method compared to previous work are: (i) the hybrid representation provides a way of combining the strengths of various methods (giving something greater than the sum of its parts); (ii) cubic splines are used to compress

[1]Admiralty charts are nautical charts issued by the United Kingdom Hydrographic Office (UKHO).

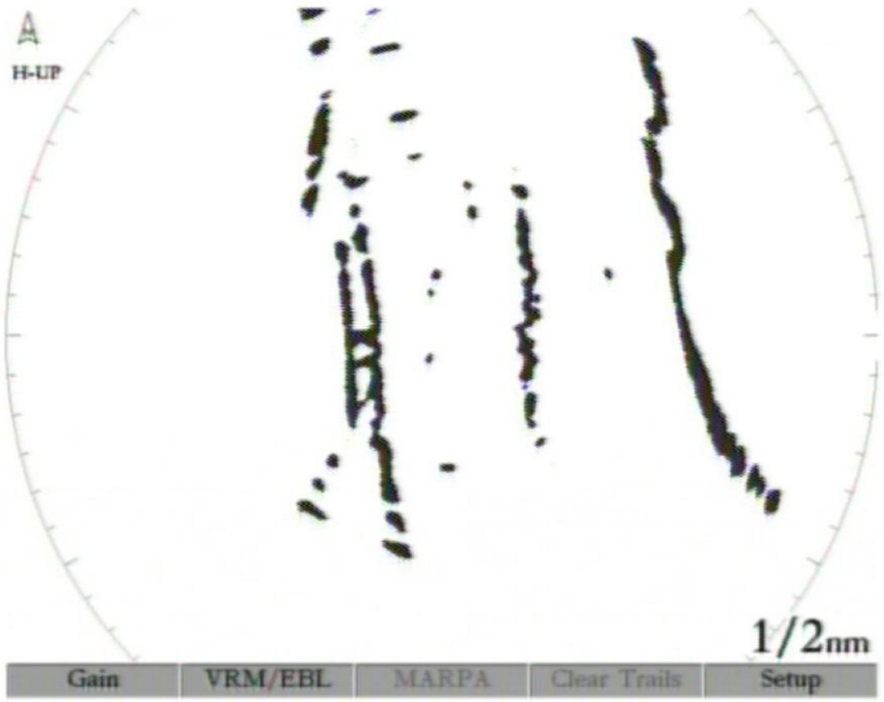


Figure 5.3: The raw RADAR data from the GMR18.

trajectories and provide a continuous representation; (iii) the spline representation allows RADAR scans to be re-rendered at sub-scan resolution, compensating for the egomotion between time-steps and (iv) the occupancy grid can be re-rendered within the temporal sliding window giving superior results compared with a filtering style solution.

The system is useful when dealing with RADAR data because: (i) intermittent measurements are fused making them clearly visible as stationary objects; (ii) clutter is rejected; (iii) dynamic objects are automatically tracked with the appropriate motion model and (iv) measurements are fused within an occupancy grid to give a clearer picture of the surrounding environment.

This chapter and the last have shown how dynamic objects can be incorporated directly within the SLAM map and how hybrid mapping techniques can be used to obtain a more efficient/sensible representation of the environment. This allows us to obtain a real-time metric estimate of the marine environment surrounding a boat. The next part of the dissertation shows how computer vision can be used to obtain

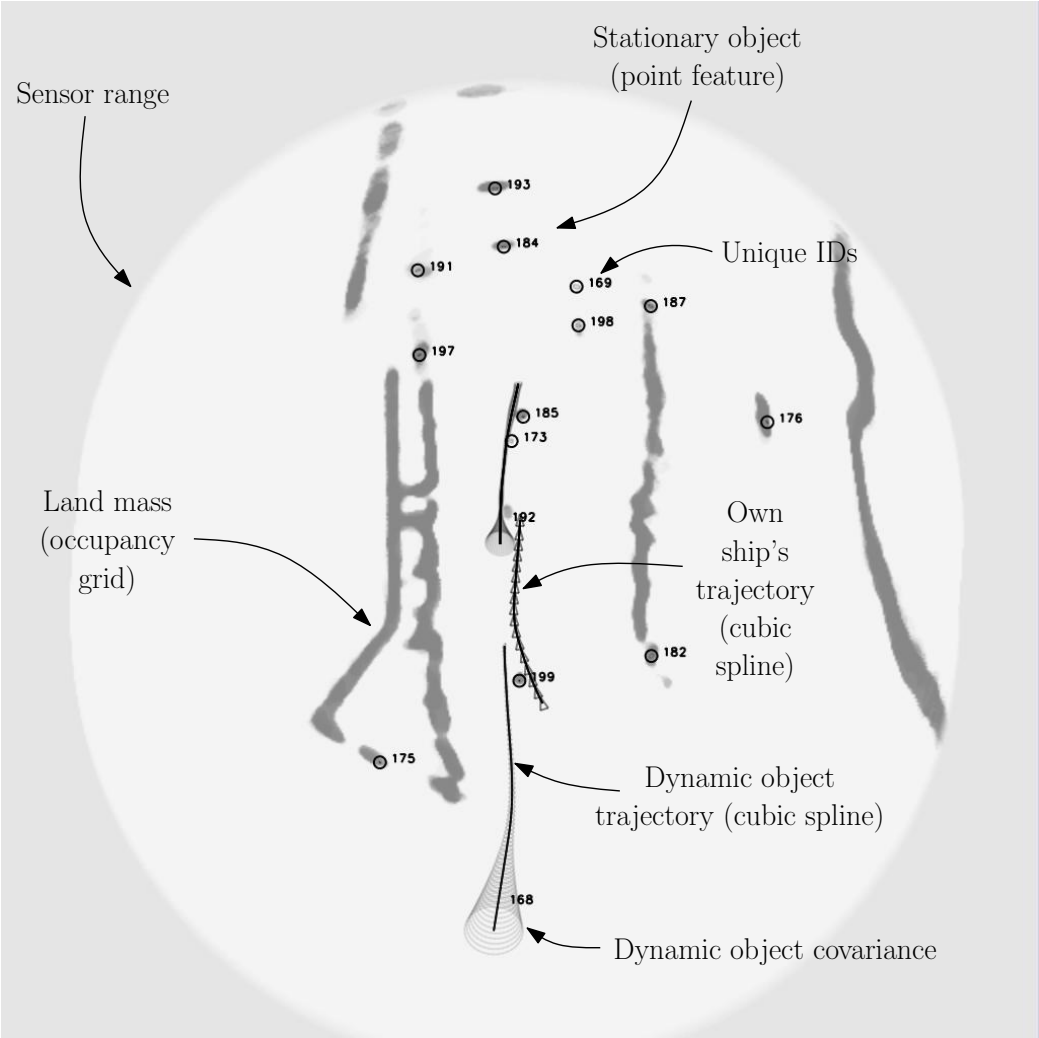


Figure 5.4: The system in action – the occupancy grid represents land masses, point features are used for stationary objects and cubic splines are used to represent the trajectories of dynamic objects.

stabilised video streams of the objects in the environment. These stabilised video streams can then be used to enrich the metric estimate as presented in this chapter, with visual information.

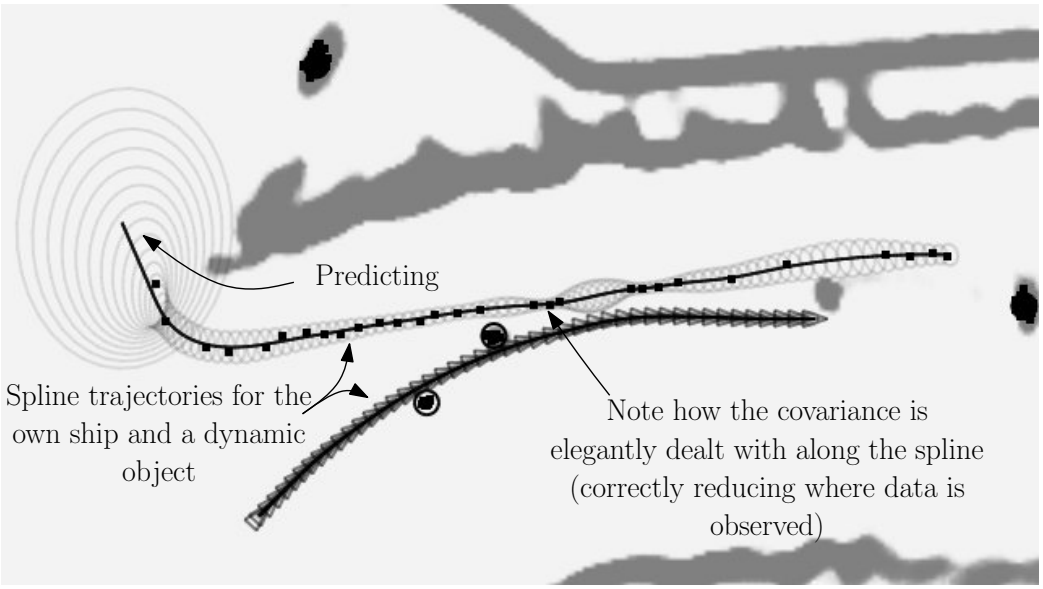


Figure 5.5: Example of spline representation in the system.

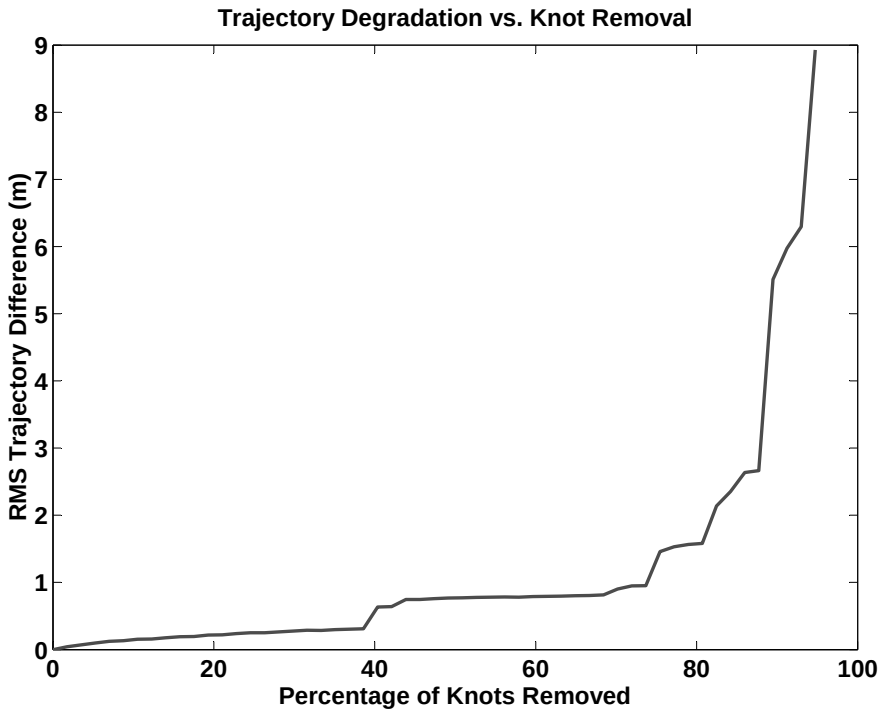


Figure 5.6: Quantitative analysis of how trajectories degrade as they are represented using fewer and fewer knots.

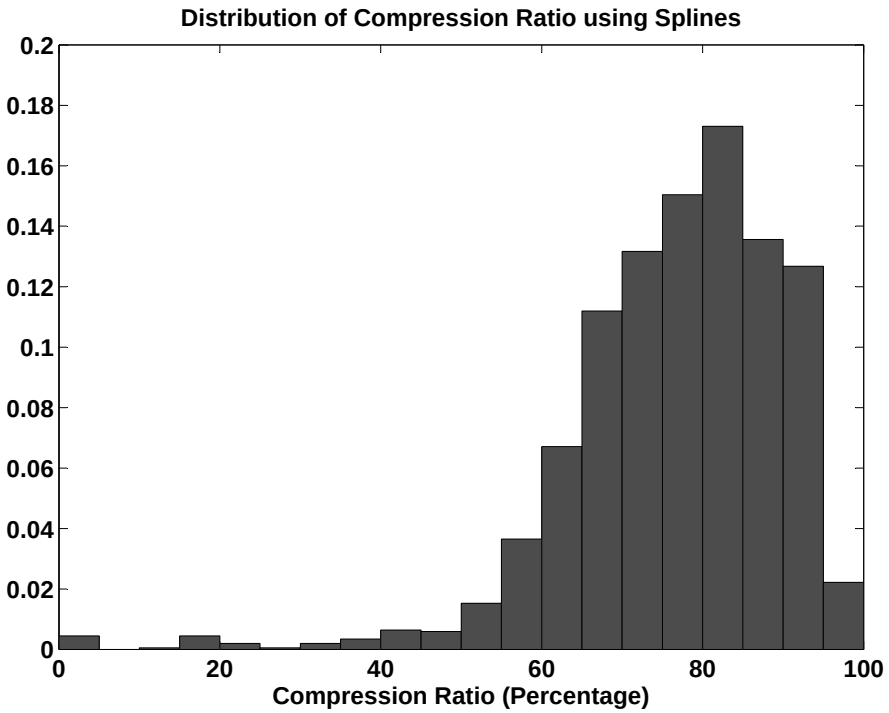


Figure 5.7: Quantitative analysis of the amount of trajectory compression achieved during the experiment.

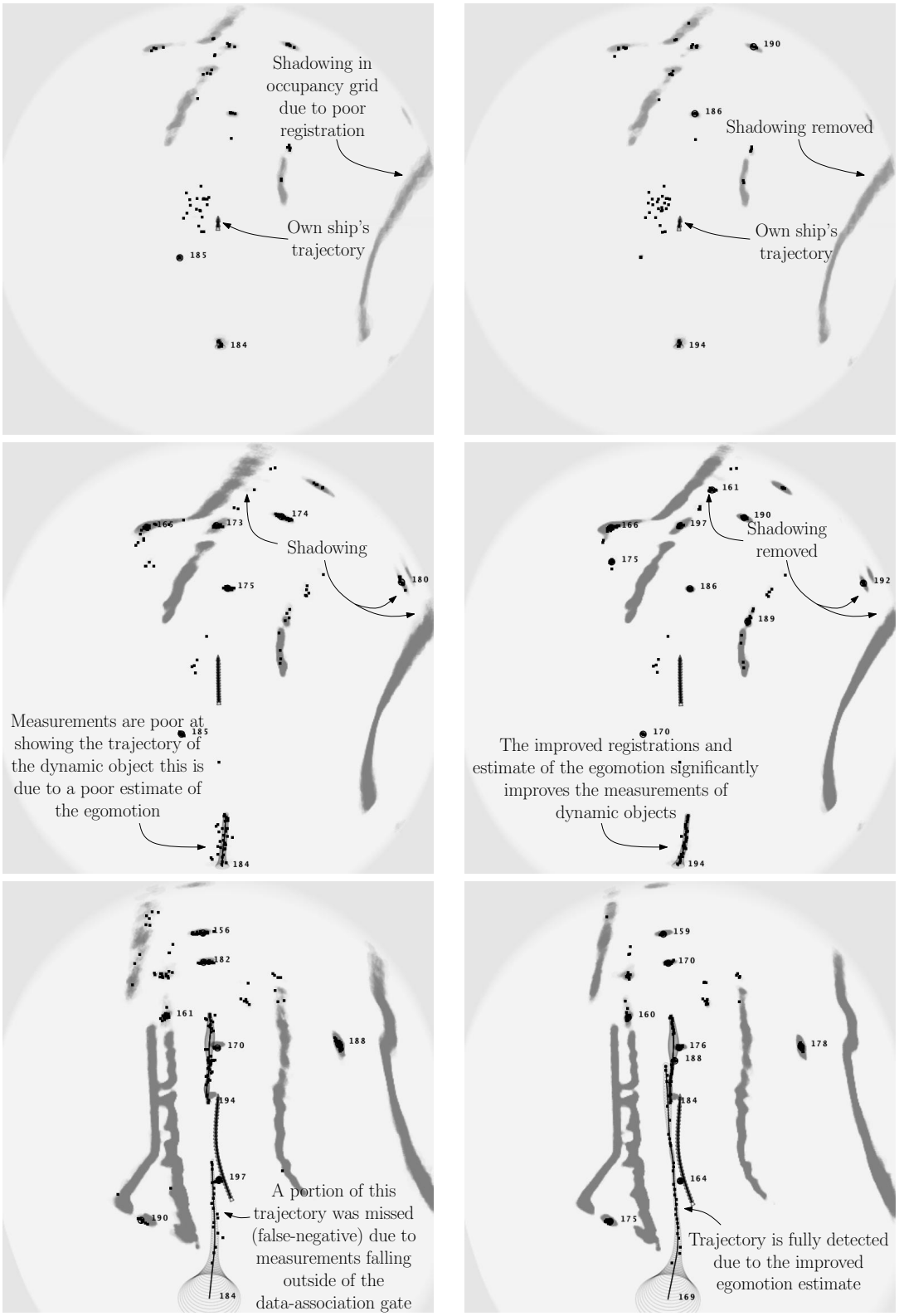


Figure 5.8: Qualitative evaluation: (left) re-rendering turned off and (right) re-rendering turned on.

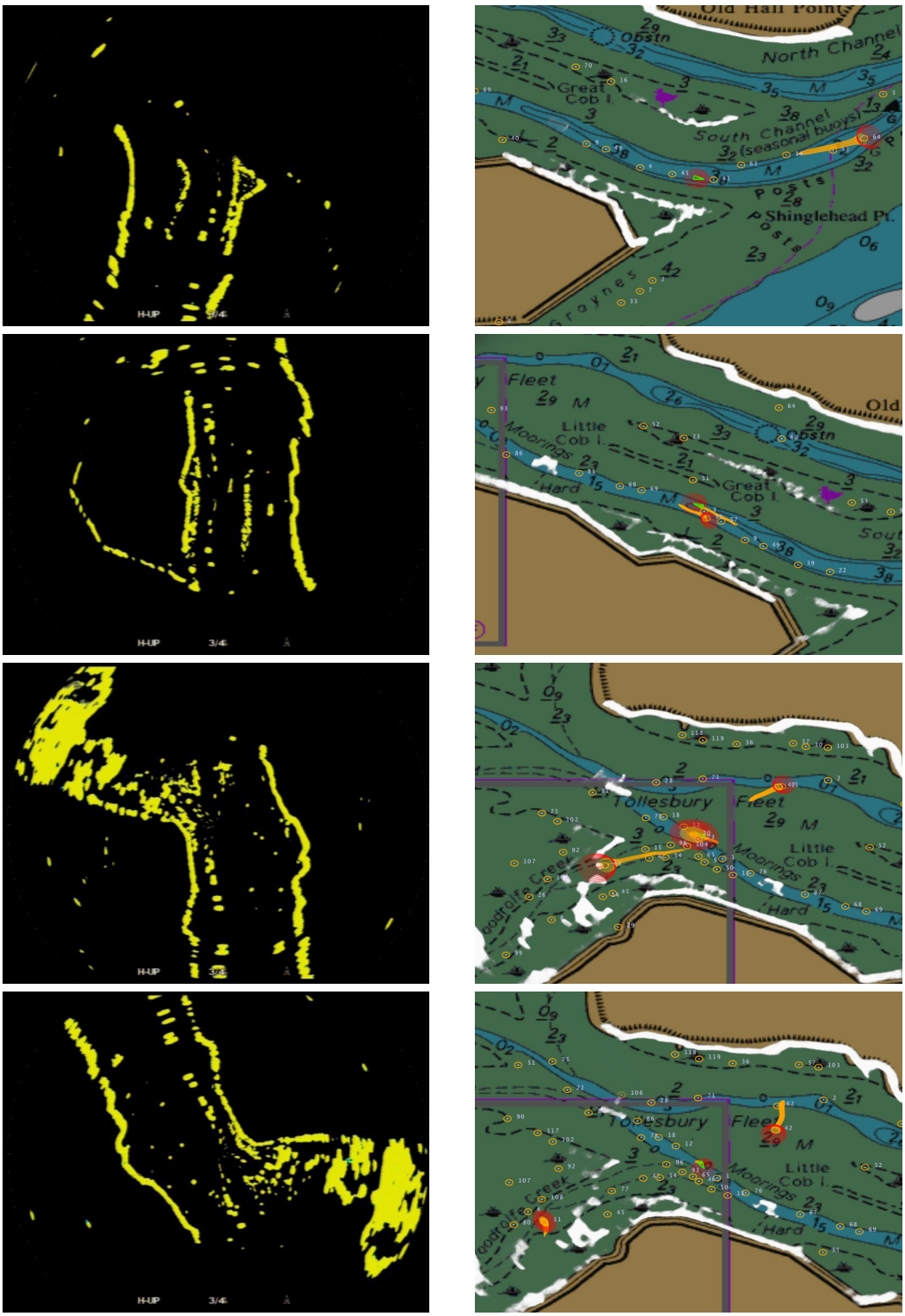


Figure 5.9: Qualitative evaluation of an 18 minute voyage: (left) raw RADAR data and (right) the output from HSLAMIDE is manually registered to a nautical chart (the top-row) at the beginning of the voyage and the alignment is still good (the bottom-row) at the end of the voyage.

Part II

Estimating Visual Information

So far we have shown how to estimate the metric aspects of the environment around a marine vessel using Hybrid SLAM in Dynamic Environments. This gives us an estimate of the positions, speeds, sizes and shapes of nearby objects and landmasses. This part of the dissertation is about visual tracking, which given a series of video frames, 'tracks' object(s) from one frame to the next, computing their position, rotation, scale and potentially other parameters. We present a new method for visual tracking that uses a region-based, level-set framework. The significant difference between our method and all others is that we use the pixel-wise posterior when computing the foreground/background pixel membership, as opposed to the traditional pixel-wise likelihood. We use the superior performance of our tracker to control a high performance pan-tilt-zoom device. Specifically, we use the computed position to control the pan-tilt axes and the computed scale to control the zoom axis. This gives us stabilised video sequences of objects in the environment, which can be used to enrich the metric estimate with visual information. We will now describe our two new methods for visual tracking, which use pixel-wise posteriors as opposed to pixel-wise likelihoods.

Chapter 6

Pixel-Wise Posterior Tracking

6.1 Introduction

Not only do we require fast and reliable visual tracking for the objective of this dissertation, but it is a prerequisite for a vast number of other applications in computer vision. Though it has been the subject of intense effort over the last two decades, it remains a difficult problem for a number of reasons. In particular, when tracking previously unseen objects, many of the constraints that give reliability to other tracking systems – such as strong prior information about shape, appearance or motion – are unavailable. For further reading on the vast quantities of work relating to visual tracking refer to [Yilmaz *et al.*, 2006] and [Moeslund *et al.*, 2006][1].

One of the earliest approaches to visual tracking, originally proposed by [Lucas & Kanade, 1981], is template tracking. It has been used extensively in one form or another for over two decades (see [Baker & Matthews, 2004] for an excellent summary). The idea behind template tracking is to extract a template (a rectangular region of pixels) in one image and to warp the template into another image so that the Sum-of-Squared Differences (SSD) between pixel values is minimised. The underlying optimisation uses image derivatives to drive the warp to a local minimum. One of the key features of using a template is that it encodes both the pixel locations and colours. This makes it possible to compute the tracking parameters to sub-pixel accuracy. A significant weakness of the approach is that an accurate template must be maintained over time, which is difficult for anything but planar objects.

[1]This survey covers the period of 2000-2006 in the application area of human motion tracking. It lists more than three hundred papers.

A significant step away from template tracking was made by [Kass *et al.*, 1988]. Rather than tracking a template corresponding to an object, they tracked the boundary between the object and the background. This was achieved using an active contour to represent the boundary. One typical use of active contours is to align them to edges in the image, by using a 1D search along the direction normal to the contour (similar to the 1D search used by [Harris, 1993]). A problem with this approach is that natural images typically contain many edges that do not belong to the object of interest, which act as distractors. This was tackled by [Isard & Blake, 1998a] by using a non-parametric method, known as particle filtering, to represent multi-modal distributions. The ability to model the distractors using multi-modal distributions added robustness to the tracker, making it possible to track objects that template tracking alone would fail on.

More recently, probability distributions have been used to model an object's appearance. This is referred to as region-based tracking and is considered more robust than templates or edges for many applications. [Comaniciu *et al.*, 2000] showed how mean-shift, a 'local' non-parametric mode finding technique, could be used to align a region to the image. The objective function used for this alignment measured the similarity between the modelled appearance distribution and the observed empirical distribution. In contrast to template tracking and active contours, an elliptical region is used and rather than taking a template or using edges, the pixel values are used to build a non-parametric distribution (a colour histogram). This non-parametric distribution is somewhat weaker than a template, in the sense that it does not contain spatial information and yet captures more information than edges alone.

A key problem with both template tracking and mean-shift is how to adapt the appearance models i.e. the template or colour histogram, over time. Both methods assume a very simplistic segmentation of the object from the background, either a rectangular or elliptical region. In order to track objects for long periods of time it is crucial that the appearance model can be adapted. In all but the simplest cases this requires a good segmentation of the object from the background, otherwise pixels that belong to the background are incorporated in an object's appearance model or vice-versa. The result is that the tracker 'drifts' away from the object over a period of time and eventually fails; this is commonly referred to as tracking drift. Although active contours have a much better notion of shape than template tracking or mean-shift, they lack a region-based representation.

One region-based technique that has shown considerable promise for its ability to per-

form tracking and segmentation within a unified framework is the use of an implicit contour (level-set) to represent the boundary of the object [Osher & Paragios, 2003; Paragios & Deriche, 2000; Goldenberg *et al.*, 2001]. As well as handling topological changes seamlessly, tracking using level-sets can be couched in a fairly standard probabilistic formulation [Cremers, 2006; Cremers *et al.*, 2007], and hence can leverage the power of Bayesian methods.

In this chapter, we present a novel probabilistic framework for combined tracking and segmentation, which, as well as capturing all of the desirable properties of level-set based tracking, is very robust and runs in a few milliseconds on standard hardware. We base our framework on a generative model of image formation that represents the image as a bag-of-pixels [Jebara, 2003]. The advantage of such a model – in common with other simpler density-based representations such as colour-histograms – is the degree of invariance to viewpoint this confers.

Like [Cremers, 2006], we derive a probabilistic, region-based, level-set framework, which comprises an optimal rigid registration, followed by a segmentation to re-segment the object and account for non-rigid deformations. Aside from issues of speed (which are not addressed in [Cremers, 2006]), there are a number of key differences between [Cremers, 2006] and our work, some of which stem from the generative model we use for image data (see Section 6.3). Firstly, our derivation gives a probabilistic interpretation to the Heaviside step function used in most region-based level-set methods [Chan & Vese, 2001; Cremers, 2006]. Secondly, given this interpretation we propose a pixel-wise posterior term, as opposed to a likelihood, which allows us to marginalise out model parameters at a pixel level. As we show in Section 6.3, this derives naturally from our generative model, and is a subtle but absolutely crucial difference between our method and others e.g. [Cremers, 2006; Paragios & Deriche, 2000; Goldenberg *et al.*, 2001], as our results show in Section 6.8. Thirdly, in contrast to [Chan & Vese, 2001; Cremers, 2006] and similar to [Freedman & Zhang, 2004; Zhang & Freedman, 2005], we assume a non-parametric distribution for image values as opposed to a single Gaussian (for an entire region). The superior performance of our method is particularly noticeable when using non-parametric distributions and much less noticeable when using smooth Gaussian distributions (as presented in [Cremers, 2006]). Finally, we introduce a prior on the embedding function which constrains it to be an approximate signed distance function. We show that this gives a clean probabilistic interpretation to the idea proposed by [Li *et al.*, 2005] and avoids the need for reinitialisation of the embedding function that is necessary in the majority

of level-set based approaches.

Our work also bears some similarity to [Yilmaz, 2007], who sought the rigid transformation that best aligns a fixed shape-kernel with image data using the Bhattacharyya coefficient. This work extended the pioneering work of this type [Comaniciu *et al.*, 2000; Collins, 2003] to handle translation+scale+rotation as opposed to translation only or translation+scale. In contrast to [Yilmaz, 2007], however, we allow the shape to change online using an implicit contour and propose a novel framework using pixel-wise posteriors, which removes the cost of building an empirical distribution and testing it with the Bhattacharyya coefficient. This has a second hidden benefit as it avoids the need to build a 'good' empirical distribution given limited data; we find in practice this gives a significant improvement over [Comaniciu *et al.*, 2000; Collins, 2003; Yilmaz, 2007].

Unlike [Cremers, 2006], [Freedman & Zhang, 2004; Zhang & Freedman, 2005] use a non-parametric distribution for image data. They derive contour flows based on both KL-divergence and the Bhattacharyya coefficient. Though they demonstrate that both are effective for tracking, they do not model rigid transformation parameters explicitly. They must recompute their non-parametric distributions at every iteration, and – as we show in Section 6.8 – objectives based on the Bhattacharyya coefficient are inferior to the one we propose.

Within our framework (and other similar work), because the segmentation is performed rapidly and reliably online, the appearance and shape models of the object can be updated over time without suffering from the significant problems of tracking drift that plague other algorithms. Our framework is general enough to be extended to various types of prior information and various imaging modalities, but in this dissertation we restrict ourselves to the problem of tracking the 2D projections of either 2D or 3D objects in ordinary colour video. In summary, the key benefits of our method are: (i) an extensible probabilistic framework; (ii) robustness - given by pixel-wise posteriors and marginalisation; (iii) real-time performance; (iv) excellent cost function characteristics; (v) no need to compute empirical distributions at every frame; (vi) online learning (i.e. adaption of appearance and shape characteristics); (vii) flexibility to track many different types of object and (viii) high invariance to view and appearance changes.

The remainder of this chapter is organised as follows: Section 6.2 describes the representation of the object being tracked; Section 6.3 derives a probabilistic framework from a simple generative model; Section 6.4 outlines the level-set segmentation; Sec-

tion 6.5 shows the registration process; Section 6.6 describes our method for dealing with tracking drift; Section 6.7 outlines the online learning process; Section 6.8 shows our results and Section 6.9 concludes with a summary and discussion.

6.2 The Representation

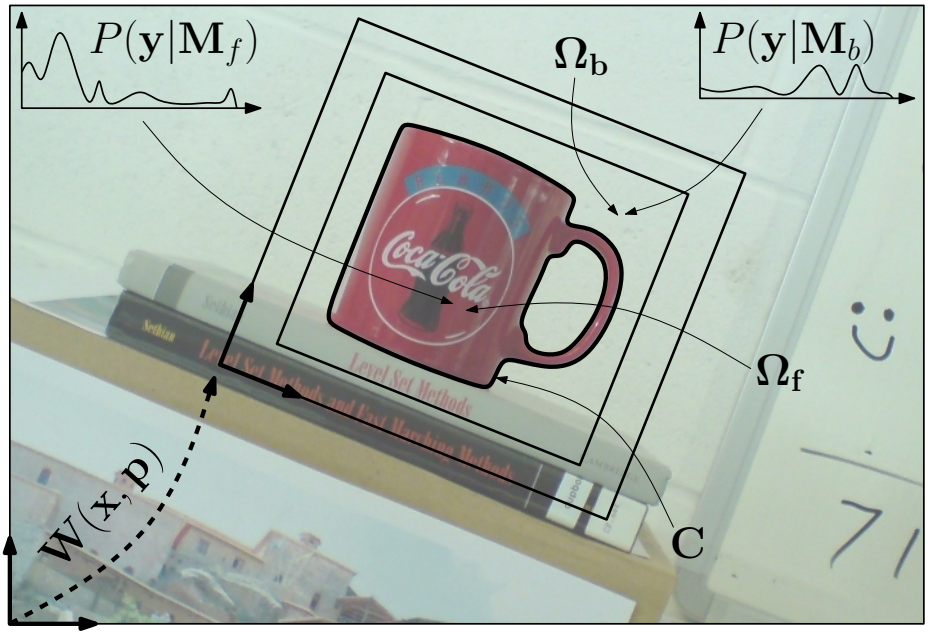


Figure 6.1: Representation of the object, showing: the contour $\mathbf{C}$, the set of foreground pixels $\mathbf{\Omega}_f$, the set of background pixels $\mathbf{\Omega}_b$, the foreground model $P(\mathbf{y}|M_f)$, the background model $P(\mathbf{y}|M_b)$ and the warp $\mathbf{W}(\mathbf{x},\mathbf{p})$.

We represent the object being tracked by its shape $\mathbf{C}$, its location in the image $\mathbf{W}(\mathbf{x},\mathbf{p})$ and two underlying appearance models: one for the foreground $P(\mathbf{y}|M_f)$ and one for the background $P(\mathbf{y}|M_b)$. Figure 6.1 illustrates this with a simple example.

Shape: is represented by the zero level-set $\mathbf{C} = \{\mathbf{x}|\mathbf{\Phi}(\mathbf{x}) = 0\}$ of an embedding function $\mathbf{\Phi}(\mathbf{x})$ [Osher & Paragios, 2003; Cremers *et al.*, 2007]. In our case, the shape $\mathbf{C}$ is a 2D contour and the embedding function $\mathbf{\Phi}(\mathbf{x})$ is a 3D function represented on a discrete grid of $\mathbf{x}$ values. Figure 6.2 shows examples of embedding functions and their corresponding level-sets for a variety of different shape contours. Given the shape contour $\mathbf{C}$, the pixels $\mathbf{\Omega}$ in the object frame are segmented into two regions: one for the foreground $\mathbf{\Omega}_f$ and one for the background $\mathbf{\Omega}_b$.

Location: is described by a warp $\mathbf{W}(\mathbf{x},\mathbf{p})$ that takes a pixel location $\mathbf{x}$ in the object frame and warps it into the image frame according to parameters $\mathbf{p}$.

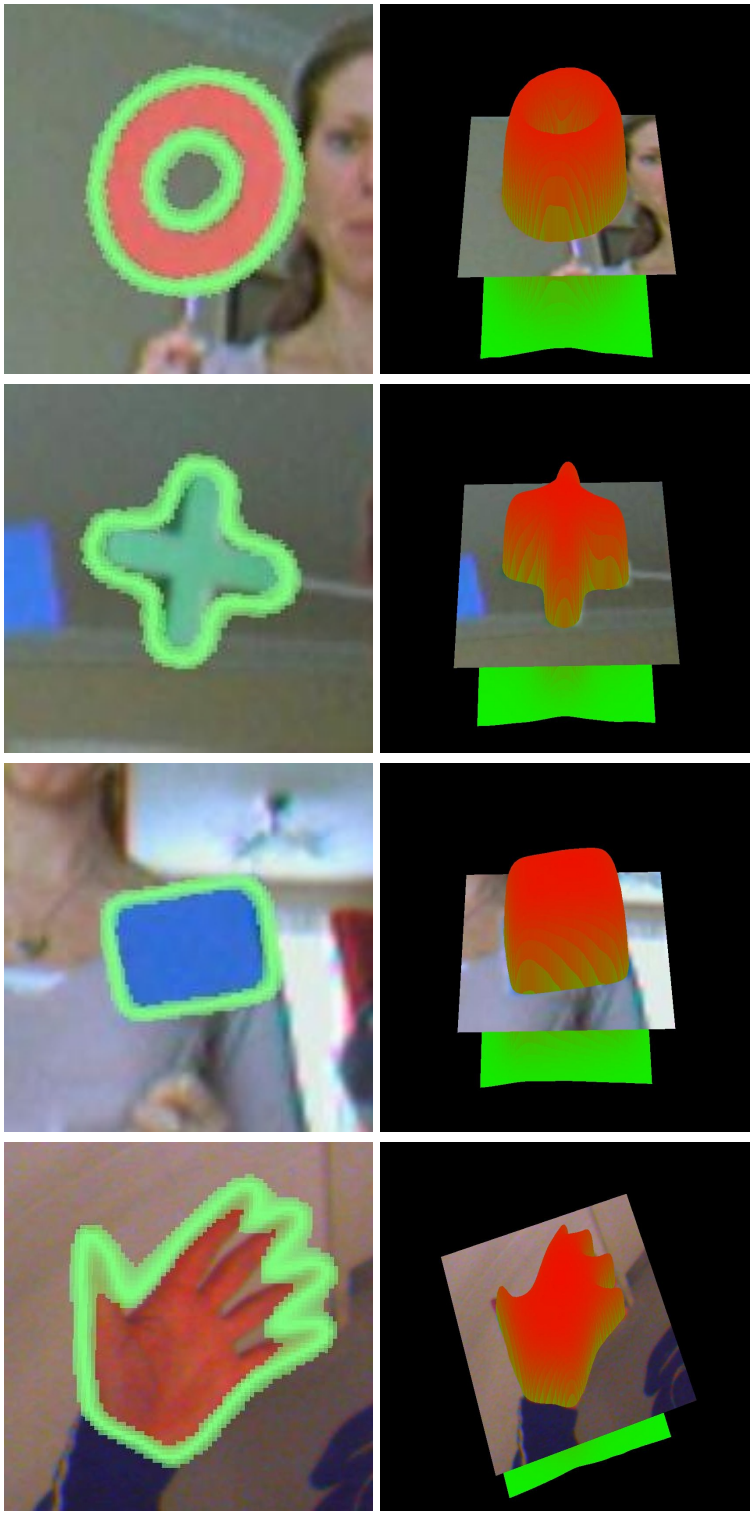


Figure 6.2: A selection of object shapes and their corresponding level-sets: (left column) shows the 2D shape contour $\mathbf{C}$ and (right column) shows the corresponding 3D embedding function $\mathbf{\Phi}(\mathbf{x})$ and the level-set $\mathbf{\Phi}(\mathbf{x}) = 0$.

Appearance models: $P(\mathbf{y}|M_f)$ and $P(\mathbf{y}|M_b)$ are represented with RGB histograms using 32 bins per channel. The histograms are initialised either from a detection module or a user inputted initial bounding box. The pixels inside the bounding box are used to build the foreground model and the pixels from an inflated bounding box are used to build the background model. The two initial distributions are then used to produce a tentative segmentation, which is in turn used to rebuild the model. This procedure is iterated until the shape converges (similar to [Rother *et al.*, 2004]). Once tracking commences, the appearance models and shape $\mathbf{C}$ are estimated (adapted) online, as described in Section 6.7.

In summary, we use the following notation:

- $\mathbf{x}$: A pixel location in the object's coordinate frame.
- $\mathbf{y}$: A pixel value (in our experiments this is a RGB value).
- $\mathbf{I}$: The image.
- $\mathbf{W}(\mathbf{x}, \mathbf{p})$: Warp with parameters $\mathbf{p}$.
- $M = \{M_f, M_b\}$: Model parameter either foreground or background.
- $P(\mathbf{y}|M_f)$: Foreground model over pixel values $\mathbf{y}$.
- $P(\mathbf{y}|M_b)$: Background model over pixel values $\mathbf{y}$.
- $\mathbf{C}$: The contour that segments the foreground from the background.
- $\mathbf{\Phi}(\mathbf{x})$: Shape kernel (in our case the level-set embedding function).
- $\mathbf{\Omega} = \{\mathbf{\Omega}_f, \mathbf{\Omega}_b\}$: Pixels in the object frame $[\{\mathbf{x}_0, \mathbf{y}_0\}, \ldots, \{\mathbf{x}_N, \mathbf{y}_N\}]$, which are partitioned into foreground pixels $\mathbf{\Omega}_f$ and background pixels $\mathbf{\Omega}_b$.
- $H_\epsilon(z)$: Smoothed Heaviside step function.
- $\delta_\epsilon(z)$: Smoothed Dirac delta function.

6.3 The Generative Model

Figure 6.3 illustrates the simple generative model we use to represent the image formation process. This model treats the image as a bag-of-pixels [Jebara, 2003] and

can, given the model M, the shape $\mathbf{\Phi}$ and the location $\mathbf{p}$, be used to sample pixels $\{\mathbf{x}, \mathbf{y}\}$.

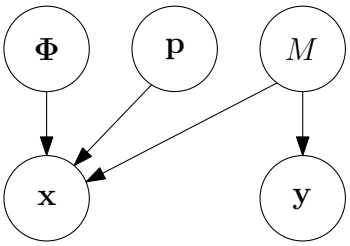


Figure 6.3: Generative model representing the image as a bag-of-pixels.

Although the resultant image would not look like the true foreground/background image to a human (the pixels would be jumbled up), the colour distributions corresponding to the foreground/background regions $\mathbf{\Omega}_f/\mathbf{\Omega}_b$ would match the models $P(\mathbf{y}|M_f)$ and $P(\mathbf{y}|M_b)$, see Figure 6.4 for an example. It is this simplicity that gives more invariance to viewpoint and allows 3D objects to be tracked robustly without having to model their specific 3D structure.

Figure 6.4: Showing the effect of using a bag-of-pixels model: (left) original image and (right) pixels sampled from the foreground and background colour distributions. Both of these images would have an equal chance of being sampled from our generative model.

The joint distribution for a single pixel given by the model in Figure 6.3 is:

$$P(\mathbf{x}, \mathbf{y}, \mathbf{\Phi}, \mathbf{p}, M) = P(\mathbf{x}|\mathbf{\Phi}, \mathbf{p}, M)P(\mathbf{y}|M)P(M)P(\mathbf{\Phi})P(\mathbf{p}). \tag{6.1}$$

We now divide (6.1) by $P(\mathbf{y}) = \sum_{i=\{f,b\}} P(\mathbf{y}|M_i)P(M_i)$ to give:

$$P(\mathbf{x}, \mathbf{\Phi}, \mathbf{p}, M|\mathbf{y}) = P(\mathbf{x}|\mathbf{\Phi}, \mathbf{p}, M)P(M|\mathbf{y})P(\mathbf{\Phi})P(\mathbf{p}), \tag{6.2}$$

where the term $P(M|\mathbf{y})$ is the pixel-wise posterior, of the model M, given a pixel value $\mathbf{y}$:

$$P(M_j|\mathbf{y}) = \frac{P(\mathbf{y}|M_j)P(M_j)}{\sum_{i=\{f,b\}} P(\mathbf{y}|M_i)P(M_i)} \qquad j = \{f, b\}. \tag{6.3}$$

Using this posterior is equivalent to applying Bayesian model-selection to each individual pixel[2]. We are interested in finding the pose $\mathbf{p}$ and the shape $\mathbf{\Phi}$, where the model parameter M can be considered a nuisance variable. The Bayesian way of dealing with nuisance variables is to marginalise them out using the sum rule. We take this approach to obtain the pixel-wise posterior of the shape $\mathbf{\Phi}$ and the location $\mathbf{p}$ given a pixel $\{\mathbf{x}, \mathbf{y}\}$:

$$P(\mathbf{\Phi}, \mathbf{p}|\mathbf{x}, \mathbf{y}) = \frac{1}{P(\mathbf{x})} \sum_{j=\{f,b\}} \{P(\mathbf{x}|\mathbf{\Phi}, \mathbf{p}, M_j)P(M_j|\mathbf{y})\} P(\mathbf{\Phi})P(\mathbf{p}). \tag{6.4}$$

Note that the pixel-wise posterior and marginalisation are the subtle but crucial differences to the work in [Cremers, 2006], which lacks the marginalisation step and uses a pixel-wise likelihood $P(\mathbf{y}|M)$. We show in Section 6.8 that our formulation yields a much better behaved objective. We consider two possible methods for fusing the pixel-wise posteriors: (i) a logarithmic opinion pool (LogOP):

$$P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega}) = \prod_{i=1}^{N} \left\{ \sum_{j=\{f,b\}} \{P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, M_j)P(M_j|\mathbf{y}_i)\} \right\} P(\mathbf{\Phi})P(\mathbf{p}) \tag{6.5}$$

and (ii) a linear opinion pool (LinOP):

[2]It is also the distribution that would be computed in the E-Step of an EM solution to the problem.

$$P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega}) = \sum_{i=1}^{N} \left\{ \sum_{j=\{f,b\}} \{P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, M_j)P(M_j|\mathbf{y}_i)\} \right\} P(\mathbf{\Phi})P(\mathbf{p}). \tag{6.6}$$

The logarithmic opinion pool is normally the preferred choice and is most similar to previous work [Cremers, 2006; Cremers *et al.*, 2007]. It assumes pixel-wise independence; whereas the linear opinion pool is equivalent to marginalising over the pixel locations – this is allowed as our bag-of-pixels generative model treats pixel locations as a random variable. We continue our derivation assuming a logarithmic opinion pool for clarity, but also include results using a linear opinion pool for completeness. For more information on opinion pools refer to [Manyika & Durrant-Whyte, 1994]. Note the term $\frac{1}{P(\mathbf{x})}$ has been dropped as it is constant for all pixel locations and we only seek to maximise $P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega})$.

6.4 Segmentation

The typical approach to region-based segmentation is to take a product of the pixel-wise likelihood functions $\prod_{i=1}^{N} P(\mathbf{I}(x_i)|M_i)$, over the pixel locations $\mathbf{x}_i$, to get the overall likelihood $P(\mathbf{I}|M)$. This can then be expressed as a summation by taking logs and optimised using variational level-sets [Osher & Paragios, 2003; Cremers *et al.*, 2007]. In contrast to these methods, our derivation leads to pixel-wise posteriors and marginalisation (6.5), a subtle but important difference.

For the remainder of this section, in order to simplify our expressions (and without loss of generality), we assume that the registration is correct and therefore $\mathbf{x}_i = \mathbf{W}(\mathbf{x}_i, \mathbf{p})$. We now specify the term $P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, M)$ in (6.5) and the term $P(M)$ in (6.3) :

$$P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, M_f) = \frac{H_\epsilon(\mathbf{\Phi}(\mathbf{x}_i))}{\eta_f} \qquad P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, M_b) = \frac{1 - H_\epsilon(\mathbf{\Phi}(\mathbf{x}_i))}{\eta_b} \tag{6.7}$$

$$P(M_f) = \frac{\eta_f}{\eta} \qquad P(M_b) = \frac{\eta_b}{\eta}, \tag{6.8}$$

where

$$\eta = \eta_f + \eta_b, \qquad \eta_f = \sum_{i=1}^{N} H_\epsilon(\mathbf{\Phi}(\mathbf{x}_i)), \qquad \eta_b = \sum_{i=1}^{N} 1 - H_\epsilon(\mathbf{\Phi}(\mathbf{x}_i)). \tag{6.9}$$

Equation (6.7) represents normalised versions of the blurred Heaviside step functions used in typical region-based level-set methods and can now be interpreted probabilistically as model specific spatial priors for a pixel location $\mathbf{x}$. Equation (6.8) represents the model priors, which are given by the ratio of the area of the model specific region to the total area of both models. Equation (6.9) contains the normalisation constants (note that $\eta = N$).

We now specify a geometric prior on $\mathbf{\Phi}$ that rewards a signed distance function:

$$P(\mathbf{\Phi}) = \prod_{i=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp - \frac{(|\nabla \mathbf{\Phi}(\mathbf{x}_i)| - 1)^2}{2\sigma^2}, \tag{6.10}$$

where σ specifies the relative weight of the prior. This gives a probabilistic interpretation to the work in [Li *et al.*, 2005]. The term $|\nabla \mathbf{\Phi}(\mathbf{x}_i)|$ is the magnitude of the gradient at point $\mathbf{x}_i$ and for a signed distance function should equal one everywhere. Maintaining a signed distance function is desirable as it has good properties when performing the registration step, which will be described in the next section. Substituting (6.7), (6.8), (6.9) and (6.10) into (6.5) and taking logs, gives the following expression for the log posterior:

$$\log(P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega})) \propto \sum_{i=1}^{N} \left\{ \log\left(P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, \mathbf{y}_i)\right) - \frac{(|\nabla \mathbf{\Phi}(\mathbf{x}_i)| - 1)^2}{2\sigma^2} \right\} + N \log\left(\frac{1}{\sigma\sqrt{2\pi}}\right) + \log(P(\mathbf{p})), \tag{6.11}$$

where

$$P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, \mathbf{y}_i) = \frac{P_f H_\epsilon(\mathbf{\Phi}) + P_b(1 - H_\epsilon(\mathbf{\Phi}))}{P_f \sum H_\epsilon(\mathbf{\Phi}) + P_b \sum (1 - H_\epsilon(\mathbf{\Phi}))}$$

and

$$P_f = P(\mathbf{y}_i|M_f), \quad P_b = P(\mathbf{y}_i|M_b), \quad \mathbf{\Phi} = \mathbf{\Phi}(\mathbf{x}_i).$$

Given that we are about to optimise with respect to $\mathbf{\Phi}$, we can drop the last two terms in (6.11) and by calculus of variations [Evans, 2002] express the first variation (Gateaux derivative) of the functional as (see Appendix C for details):

$$\frac{\partial \log(P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega}))}{\partial \mathbf{\Phi}} = \frac{\delta_\epsilon(\mathbf{\Phi})(P_f - P_b)}{P_f H_\epsilon(\mathbf{\Phi}) + P_b(1 - H_\epsilon(\mathbf{\Phi}))} - \frac{1}{\sigma^2}\left[\nabla^2 \mathbf{\Phi} - \text{div}\left(\frac{\nabla \mathbf{\Phi}}{|\nabla \mathbf{\Phi}|}\right)\right], \tag{6.12}$$

where ∇^2 is the Laplacian operator and $\delta_\epsilon(\mathbf{\Phi})$ is the derivative of the blurred Heaviside step function, i.e. a blurred Dirac delta function. Interestingly, $\delta_\epsilon(\mathbf{\Phi})$ can now be interpreted as a way of expressing uncertainty on the contour $\mathbf{C}$. If we were to use Gaussian uncertainty for the contour, then the region-based uncertainty would be expressed in terms of $\text{erf}(\mathbf{\Phi})$ instead of $H_\epsilon(\mathbf{\Phi})$ (we do not explore this any further in this dissertation). We seek $\frac{\partial \log(P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega}))}{\partial \mathbf{\Phi}} = 0$ by carrying out steepest-ascent using the following gradient flow:

$$\frac{\partial \mathbf{\Phi}}{\partial t} = \frac{\partial \log(P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega}))}{\partial \mathbf{\Phi}}. \tag{6.13}$$

In practice this is implemented using a simple numerical scheme on a discrete grid. All spatial derivatives are computed using central differences and the Laplacian uses a 3×3 spatial kernel. We use $\sigma = \sqrt{50}$ and a time-step $\Delta t = 1$ for all experiments. For stability $\frac{\Delta t}{\sigma^2} < 0.25$ must be satisfied (see [Li *et al.*, 2005] for details).

6.5 Registration

It is possible to pose the tracking problem directly in a segmentation framework [Freedman & Zhang, 2004]. Instead, like [Cremers, 2006] we model the frame-to-frame registration explicitly, by having the level-set in the object frame and introducing a warp $\mathbf{W}(\mathbf{x}, \mathbf{p})$ into (6.11). The main benefits of this approach are: (i) control

over the interaction between registration (tracking) and segmentation (local shape deformation); (ii) by registering the embedding function first, fewer iterations are required to take account of shape changes (in fact we find one per frame is adequate for our sequences). We now drop any terms in (6.11) that are not a function of $\mathbf{p}$ in preparation for differentiation:

$$\log(P(\mathbf{\Phi}, \mathbf{p}|\mathbf{\Omega})) \propto \sum_{i=1}^{N} \Big\{ \log\left(P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, \mathbf{y}_i)\right) \Big\} + \log(P(\mathbf{p})) + const. \tag{6.14}$$

If we now take $P(\mathbf{p})$ to be the uninformative uniform distribution (i.e. no motion model) and compute the MAP estimate by maximising with respect to $\mathbf{p}$, then we can write:

$$\mathbf{p} = \arg\max_{\mathbf{p}} \left\{ \sum_{i=1}^{N} \log P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, \mathbf{y}_i) \right\}. \tag{6.15}$$

Using the short-hand $P(\ldots) = P(\mathbf{x}_i|\mathbf{\Phi}, \mathbf{p}, \mathbf{y}_i)$ we can differentiate once with respect to $\mathbf{p}$ to obtain the Jacobian:

$$\mathbf{G} = \frac{\partial \log P(\ldots)}{\partial \mathbf{p}} = \frac{1}{P(\ldots)} \frac{\partial P(\ldots)}{\partial \mathbf{p}} \tag{6.16}$$

and a second time to obtain the Hessian:

$$\frac{\partial \mathbf{G}}{\partial \mathbf{p}} = \frac{-1}{P(\ldots)^2} \frac{\partial P(\ldots)}{\partial \mathbf{p}}^T \frac{\partial P(\ldots)}{\partial \mathbf{p}} + \frac{1}{P(\ldots)} \frac{\partial^2 P(\ldots)}{\partial \mathbf{p}^2}. \tag{6.17}$$

Referring back to (6.4) we can break the term $\frac{\partial P(\ldots)}{\partial \mathbf{p}}$ into two components:

$$\frac{\partial P(\ldots)}{\partial \mathbf{p}} = \mathbf{J}_n B_n, \tag{6.18}$$

where

$$\mathbf{J}_n = \frac{\partial H_\epsilon}{\partial \mathbf{\Phi}} \frac{\partial \mathbf{\Phi}}{\partial \mathbf{W}} \frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \delta_\epsilon(\mathbf{\Phi}) \nabla \mathbf{\Phi}(\mathbf{x}_n) \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \tag{6.19}$$

and

$$B_n = \frac{P_f - P_b}{P_f H_\epsilon(\mathbf{\Phi}) + P_b(1 - H_\epsilon(\mathbf{\Phi}))}. \tag{6.20}$$

The second term in (6.17) will contain second-order derivatives of the level-set embedding function $\mathbf{\Phi}(\mathbf{x}_n)$. Given that this function is approximately a signed distance function due to (6.10) i.e. zero curvature near the contour, we can approximate (6.17) by its first term only. This enables us to use an approximate[3] version of the second-order Newton optimisation scheme:

$$\Delta \mathbf{p} = \left[\sum_{i=1}^{N} B_i^2 \mathbf{J}_i^T \mathbf{J}_i \right]^{-1} \sum_{i=1}^{N} \mathbf{J}_i^T B_i. \tag{6.21}$$

Equation (6.21) is then used to update the parameters $\mathbf{p}$ by composing $\mathbf{W}(\mathbf{x}_i, \mathbf{p})$ with $\mathbf{W}(\mathbf{x}_i, \Delta\mathbf{p})^{-1}$, analogous to inverse compositional tracking [Baker & Matthews, 2004]. For this reason the warp must form a group [Baker & Matthews, 2004]; however, this is acceptable as many common useful transformations in computer vision do form groups, for instance: translation, translation+scale, similarity transforms, affine transforms and homographies.

6.6 Spatial Drift Correction

Having the object represented by its location $\mathbf{p}$ and shape $\mathbf{\Phi}$ leaves an ambiguity where it is possible to explain rigid transformations of the shape either with $\mathbf{p}$ or $\mathbf{\Phi}$. Ideally, any rigid motion would be explained solely by $\mathbf{p}$; however, over time the shape $\mathbf{\Phi}$ slowly incorporates a rigid transformation. We tackle this problem by keeping the top, bottom, left and right borders[4] (B_t, B_b, B_l, B_r) balanced and the

[3] This would be exact if we perfectly maintained a signed distance function.

[4] Smallest distances between the contour and the corresponding side of the foreground box.

minimum border distance equal to four pixels. This keeps the object contour at an appropriate size and centred within the foreground box. We use the following proportional controllers:

$$\begin{aligned} T_x &= \max(-L_t, \min(K_t(B_l - B_r), L_t)) \\ T_y &= \max(-L_t, \min(K_t(B_t - B_b), L_t)) \\ S &= \max(-L_s, \min(K_s(B_{des} - B_{min}), L_s)) \end{aligned} \tag{6.22}$$

where

$$B_{min} = \min(B_l, B_r, B_t, B_b)$$

to build a drift correction warp:

$$\mathbf{W}_{dc} = \begin{bmatrix} 1+S & 0 & T_x \\ 0 & 1+S & T_y \\ 0 & 0 & 1 \end{bmatrix}. \tag{6.23}$$

This warp is applied to the level-set $\mathbf{\Phi}$ and the pose parameters $\mathbf{p}$ to compensate for spatial drift. The parameters $L_t = 0.4$ and $L_s = 0.1$ are the saturation limits, $K_t = 1$ and $K_s = 0.005$ are the gains for the translation and scale controllers respectively and $B_{des} = 4$ is the desired minimum border distance. Figure 6.5 gives an example of how the drift correction works.

6.7 Online Learning

Once registration and segmentation are completed, both the foreground and background models are adapted online. This is achieved using linear opinion pools with variable learning rates α_i, $i = \{f, b\}$:

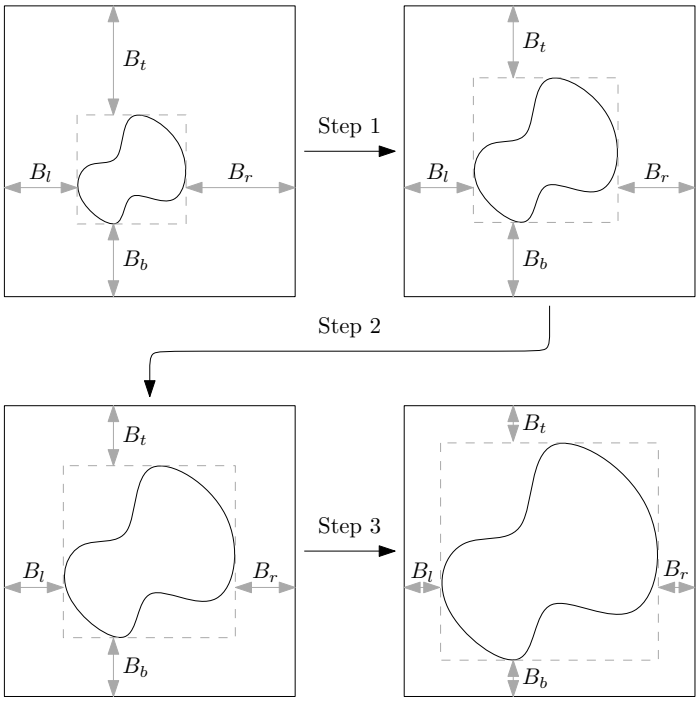


Figure 6.5: Diagram showing how the spatial drift correction works. Starting with the top-left arrangement, three warps are applied. This gradually equalises the borders and makes them equal to B_{des}.

$$P_t(\mathbf{y}|M_i) = (1-\alpha_i)P_{t-1}(\mathbf{y}|M_i) + \alpha_i P_t(\mathbf{y}|M_i), \qquad i = \{f, b\}. \tag{6.24}$$

In all experiments $\alpha_f = 0.02$ and $\alpha_b = 0.025$. For shape adaptation we control the evolution rate of the level-set using the time-step Δt.

6.8 Results

We have tested our system extensively on live video and on a variety of recorded sequences, which include objects that exhibit rapid and agile motion with significant motion blur, varying lighting, moving cameras, and cluttered and changing backgrounds. Figure 6.6 shows a qualitative evaluation of our method on three sequences. The first is a speedboat undergoing a 180° out-of-plane rotation – note how the shape is adapted online. The second is a person jumping around – note the motion blur and shape adaptation. Finally, the third is a hand being tracked from a head mounted camera past a challenging background that has a similar appearance to the object.

Figures 6.10-6.14 show a selection of different types of objects that we have successfully tracked.

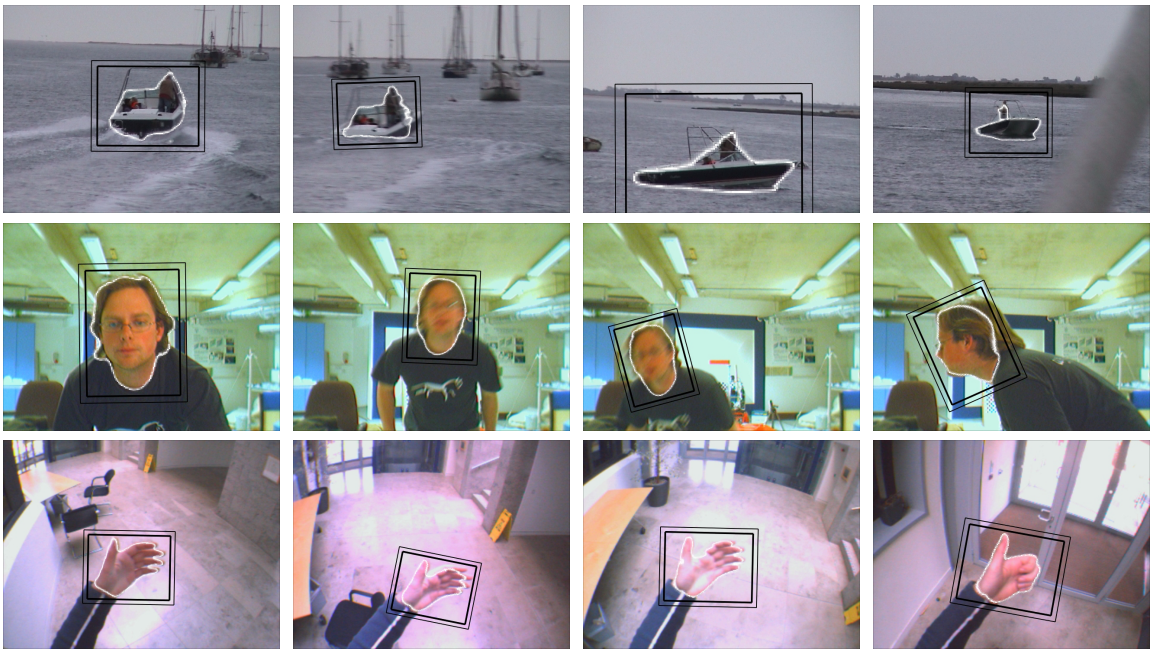


Figure 6.6: Qualitative evaluation: (top) a speedboat undergoing a 180° out-of-plane rotation illustrating shape adaptation; (middle) a person jumping around with significant motion blur and (bottom) a hand being tracked in front of a challenging background.

To perform a quantitative evaluation we have analysed the characteristics of the underlying cost function for our technique and compared this with competing alternatives on a set of pre-recorded video sequences. Figure 6.7 shows still images taken mid-sequence from a subset of these sequences; the minimum length is 400 frames and the total number of frames is over 20,000. To facilitate visualisation of the results we use a 2D rigid transformation + scale, considering each of the four dimensions separately. The competing cost functions considered correspond to the following alternative methods of tracking: level-set methods based on likelihoods [Cremers, 2006; Paragios & Deriche, 2000], mean-shift [Comaniciu *et al.*, 2000; Collins, 2003; Yilmaz, 2007], inverse compositional [Baker & Matthews, 2004] and distribution based tracking [Freedman & Zhang, 2004; Zhang & Freedman, 2005].

A good cost function has a single extremum at the true location. A poor one has multiple extrema and any local optimisation technique is liable to fall into one of these, which in practice is often the start of tracking failure. For each video frame and each dimension (translation in x and y, rotation and scale) we compute the objectives for

Figure 6.7: A selection of video frames from the data sets: (1st row) lifeboat, Coca-Cola mug, a face and a hand filmed from a head mounted camera and (2nd row) a hand using a mouse, a speedboat, a person from the caviar data set [Fisher, 2004] and a tractor. The white contour indicates the current segmentation and the two black boxes indicate the object's coordinate frame.

the competing cost functions at 40 evenly spaced points over an interval centred at the true state. We then extract all local extrema from these objectives and examine how they are distributed across the interval. To summarise this information we compute a distribution for each dimension and each cost function, using our collection of over 20,000 frames. Figure 6.8 shows a diagram of how the distribution of extrema is generated. The ideal distribution would be a delta function centred on the true state i.e. no chance of a local extremum away from the true optimum; whereas a good distribution would be peaky around the true state and have low probability of local extrema within the region it will be required to converge from. A bad distribution would be relatively flat with high probability of local extrema over the entire space, such as the one illustrated in Figure 6.8. The particular cost functions we consider are:

- **LogPWP:** Pixel-wise posteriors fused using a logarithmic opinion pool.
- **LinPWP:** Pixel-wise posteriors fused using a linear opinion pool.
- **LogLike:** Log likelihood, used in most level-set work [Cremers *et al.*, 2007; Cremers, 2006; Paragios & Deriche, 2000].
- **BhattF:** Bhattacharyya coefficient:
  $B(\boldsymbol{\Omega}_f) = \sum_{j=1}^{V} \sqrt{P(\mathbf{y}_j|M_f)P(\mathbf{y}_j|\boldsymbol{\Omega}_f)}$, used by [Comaniciu *et al.*, 2000; Collins, 2003; Yilmaz, 2007].

- **BhattFB:** Bhattacharyya coefficient with a background model: $B(\mathbf{\Omega}_f, \mathbf{\Omega}_b) = \sum_{j=1}^{V} \sqrt{P(\mathbf{y}_j|M_f)p(\mathbf{y}_j|\mathbf{\Omega}_f)} + \sum_{j=1}^{V} \sqrt{P(\mathbf{y}_j|M_b)P(\mathbf{y}_j|\mathbf{\Omega}_b)}$.

- **BhattFBM:** Bhattacharyya coefficient with a background mismatch: $B(\mathbf{\Omega}_f, \mathbf{\Omega}_b) = \sum_{j=1}^{V} \sqrt{P(\mathbf{y}|M_f)p(\mathbf{y};\mathbf{\Omega}_f)} - \sum_{j=1}^{V} \sqrt{P(\mathbf{y}_j|M_f)P(\mathbf{y}_j|\mathbf{\Omega}_b)}$, suggested by [Zhang & Freedman, 2005].

- **Ideal SSD:** Sum of squared pixel differences using the ideal template i.e. the template extracted at the current location $\mathbf{p}$. This is essentially what you would get if you had the perfect generative model giving the true pixel value at each pixel location, including the noise. This of course is *never* going to be achievable, but has been included as a useful benchmark and gives an indication of what effect incorporating texture may have.

Note:- V is the number of pixel values i.e. $32 \times 32 \times 32$; $P(\mathbf{y}|\mathbf{\Omega}_i) \quad i = \{f, b\}$ is the empirical density built from the pixels $\mathbf{\Omega}_i$ and when computing Bhattacharyya coefficients we weight the contribution of each pixel according to our shape kernel, which is identical to Yilmaz's work [Yilmaz, 2007].

Figure 6.9 shows distributions generated from over 20,000 real video frames for: translation in x, translation in y, scale and rotation.

- **Translation in x and y:** Our method has narrower distributions near the true state than all methods apart from ideal SSD and is significantly better than the log likelihood used by [Cremers, 2006]. Unlike the other methods, it also exhibits virtually no extrema outside a ± 5 pixel region – this means that our method will converge to within ± 5 pixels of the true state from anywhere within the ± 20 pixel space we have evaluated.

- **Scale:** The Bhattacharyya method and Bhattacharyya with background mismatch both have poor localisation in scale, which is in agreement with the findings of many authors. The log likelihood also poorly localises scale compared with our pixel-wise posterior based methods.

- **Rotation:** All Bhattacharyya methods and the log likelihood are poor at correctly localising the rotation. The straight Bhattacharyya coefficient for example has more than a 1% chance of exhibiting extrema anywhere in the rotation space, at a 30Hz frame rate this corresponds to approximately 1 frame in every 3 seconds of video. It is worth noting that the side lobes (at approximately

25°) exhibited by our methods and ideal SSD, are due to the self similarity corresponding to fingers in the hand sequences.

Experimentally we were unable to make the log likelihood successfully track several of our sequences, which is confirmed by its poor performance in Figure 6.9. One possible explanation is that in other work [Cremers, 2006; Cremers *et al.*, 2007; Paragios & Deriche, 2000], a single Gaussian parametric model is used. This implicitly enforces a smooth, unimodal distribution for the joint likelihood. Non-parametric representations do not exhibit these properties; however, they are better at describing complicated distributions and therefore desirable. The reason that our method can deal with these distributions is because of the normalising denominator in (6.3) and the marginalisation step in (6.4). These two steps prevent individual pixels from dominating the cost function, hence making it smoother and more well-behaved.

The work of [Freedman & Zhang, 2004] and its subsequent improvement [Zhang & Freedman, 2005] use distribution matching techniques to incorporate non-parametric distributions into a level-set framework. These methods, similar to the Bhattacharyya based methods, involve computing the empirical densities at every iteration of the optimisation, whereas our method avoids this extra cost. Not only is our method superior to these approaches in terms of cost functions (see Figure 6.9), but it is computationally cheaper to evaluate as it does not require empirical distributions. This is a significant benefit because it not only reduces the cost per iteration, but avoids the issue of having to build 'good' distributions. One explanation for the difference between the performance of these methods and ours, is that it is hard to build 'good' empirical distributions in real-time and most methods rely on simple histograms. Although this could be improved with Parzen or NP windowing techniques [Kadir & Brady, 2005], it would almost certainly sacrifice real-time performance.

Timing

All terms in (6.21) include $\delta_\epsilon(\mathbf{\Phi}(\mathbf{x}_i))$ (blurred Dirac delta function). This means that an individual pixel's contribution to the optimisation diminishes the further from the contour it is. An efficient implementation, therefore, recognises this. Our implementation ignores pixels outside a narrow band and for an object size of 180×180 runs in $500\mu s$ on a P4 3.6GHz machine. On average the system runs at a frame rate of 85Hz for the complete algorithm and if shape and appearance learning are turned off (i.e. rigid registration only) it averages 230Hz.

6.9 Conclusions

We have proposed a novel probabilistic framework for robust, real-time, visual tracking of previously unseen objects from a moving camera. The key contribution of our method and reason for its superior performance compared with others, is the use of pixel-wise posteriors as opposed to a product over pixel-wise likelihoods. In contrast to other methods [Cremers, 2006; Cremers *et al.*, 2007], we solve the registration using Gauss Newton, which has significant practical benefits, namely: (i) the difficulty associated with step size selection is removed and (ii) reliable and fast convergence. We have demonstrated the benefits of our method both qualitatively and quantitatively, with a thorough analysis of pixel-wise posteriors versus competing alternatives using over 20,000 video frames. Our results demonstrate that using pixel-wise posteriors provides excellent performance when incorporating non-parametric distributions into region based level-sets. It not only offers superior cost functions, but avoids the need for computing empirical distributions [Comaniciu *et al.*, 2000; Freedman & Zhang, 2004; Zhang & Freedman, 2005; Yilmaz, 2007] and is therefore faster.

One of the failure modes of this tracker is if an object with similar appearance occludes/interacts with the object being tracked. The result is the tracker will often be seduced away from the true object and either fail completely or end up tracking the wrong object. The next chapter describes how a more complicated generative model can be used to deal with multiple interacting objects that occlude each other.

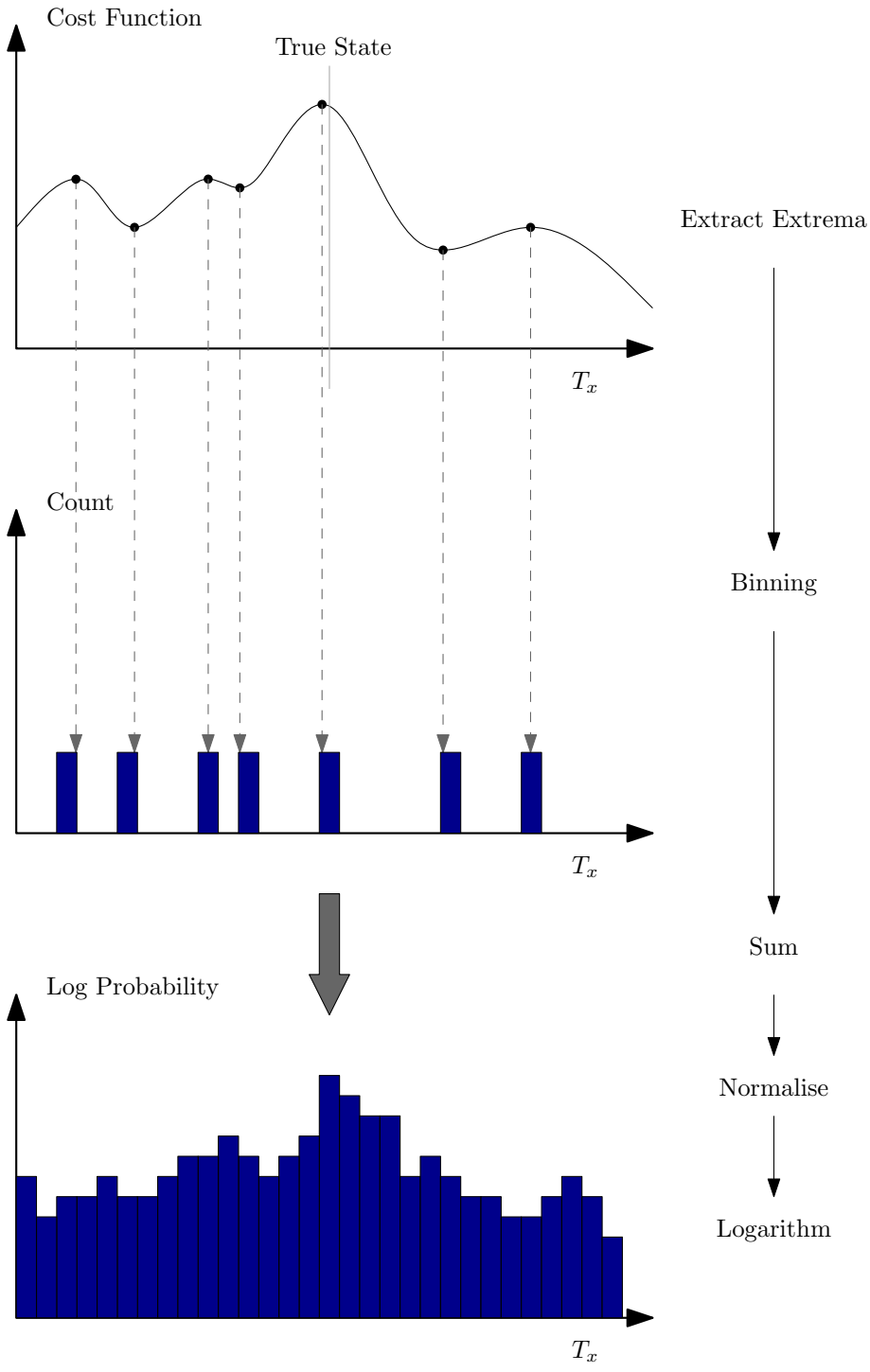


Figure 6.8: A diagram showing how the distribution of extrema is generated. For each video frame all extrema are extracted from the cost function and binned into 40 evenly spaced bins over an interval centred at the true state. These are then accumulated over all video frames into a normalised distribution. Finally, we take the log of this distribution to make visualisation easier.

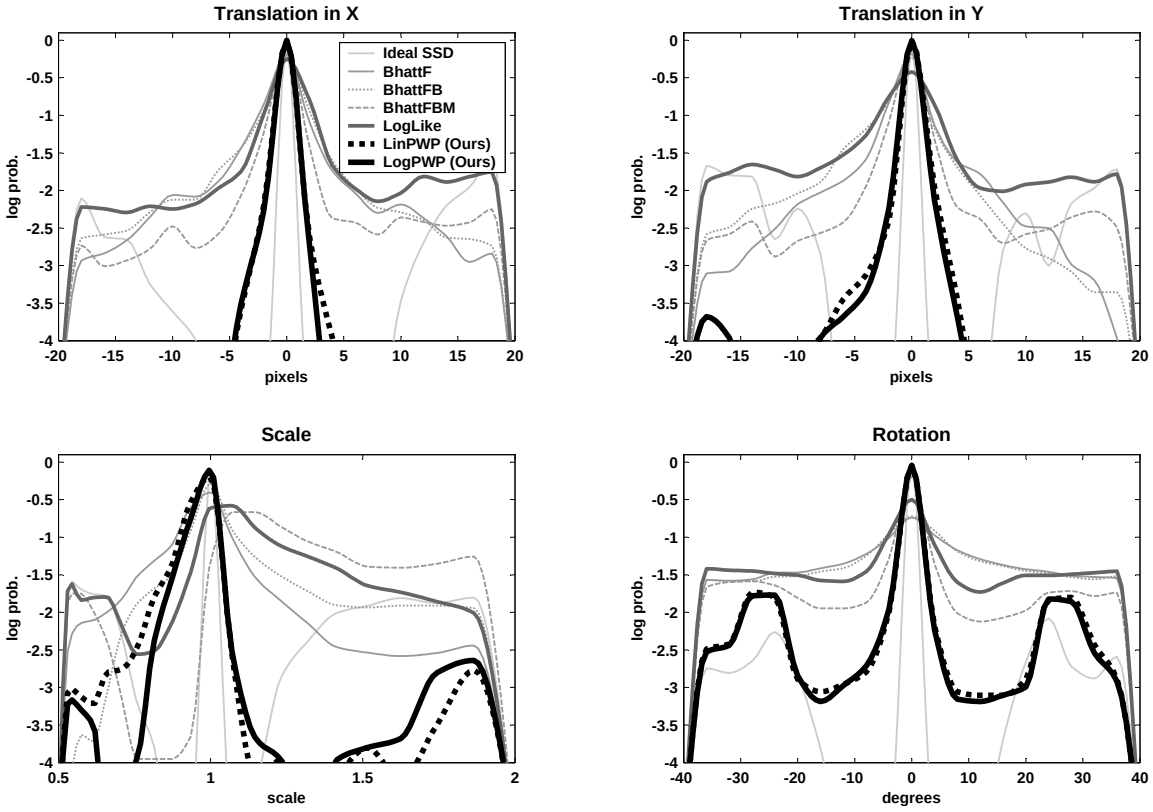


Figure 6.9: Quantitative Analysis: log probability distribution of extrema in the cost functions generated from 20,000 frames of real video data.

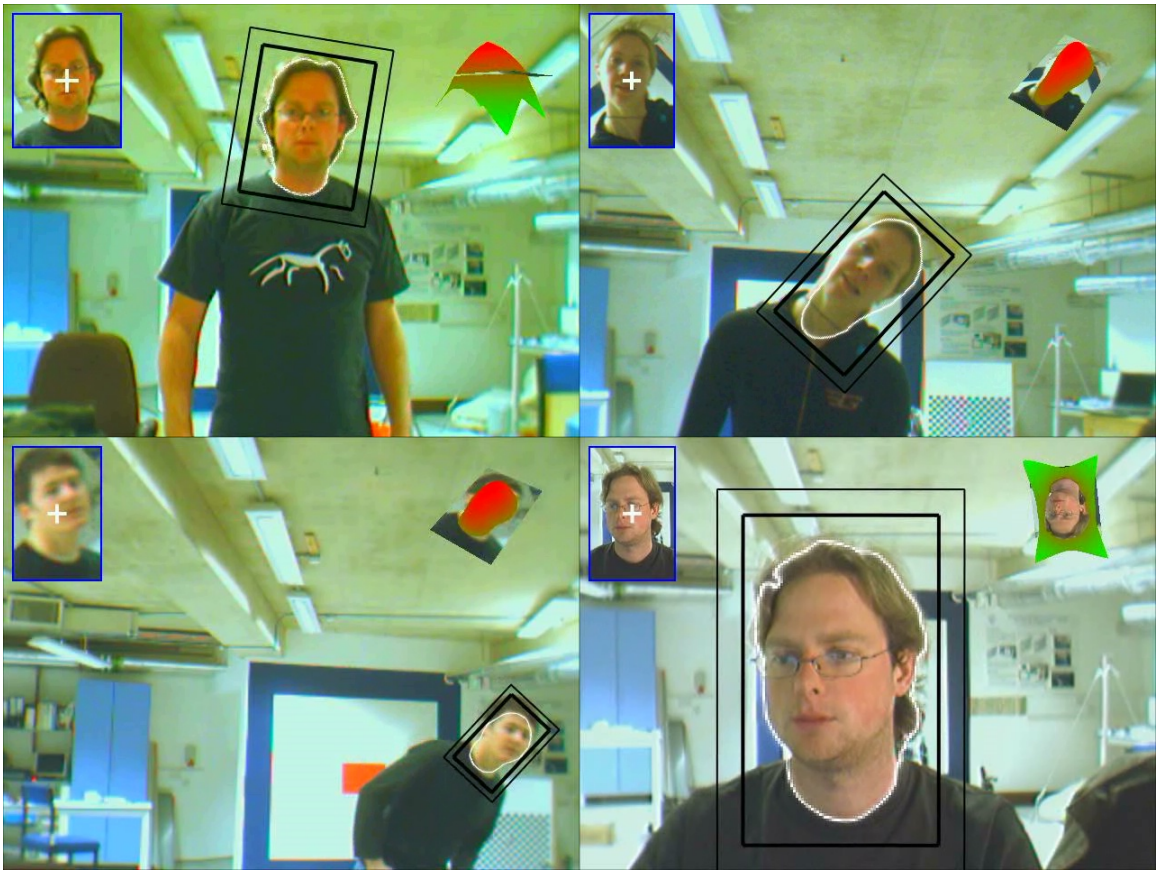


Figure 6.10: Tracking faces montage.

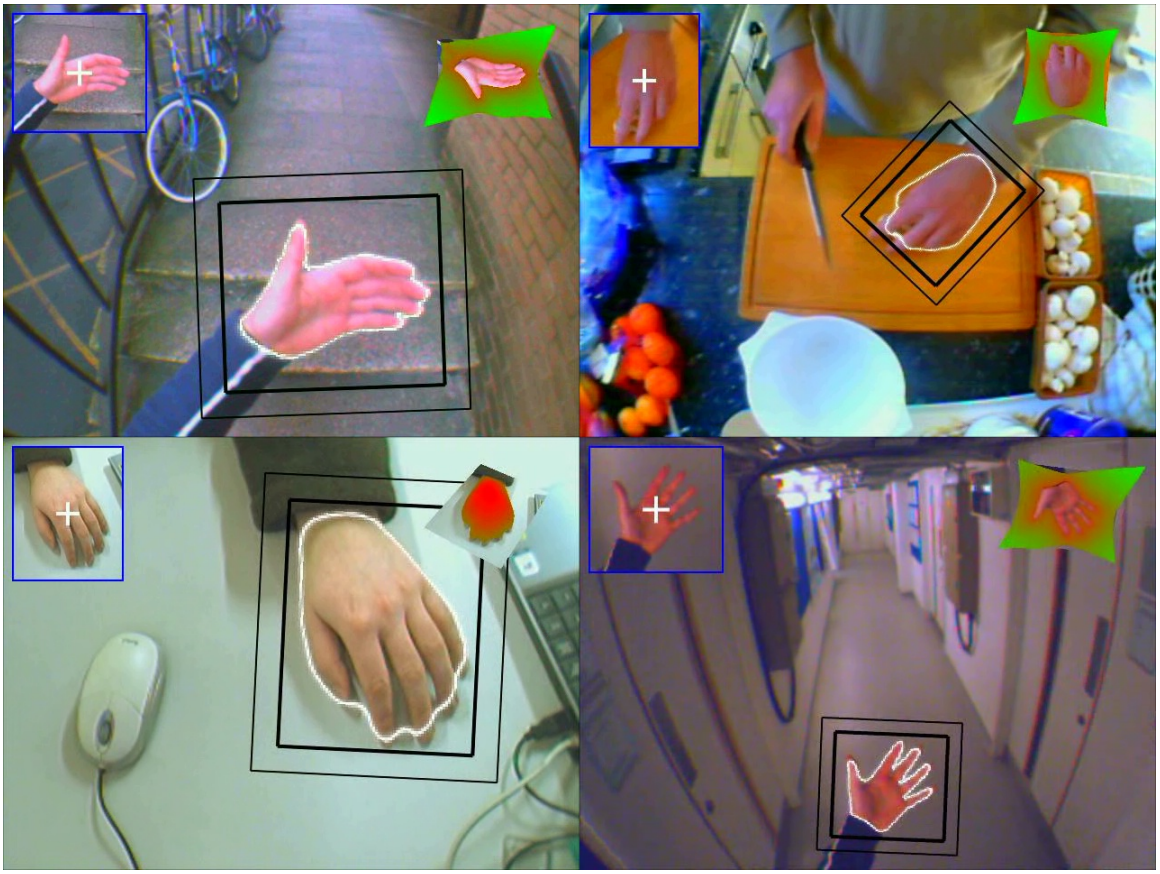


Figure 6.11: Tracking hands montage.

Figure 6.12: Tracking people montage.

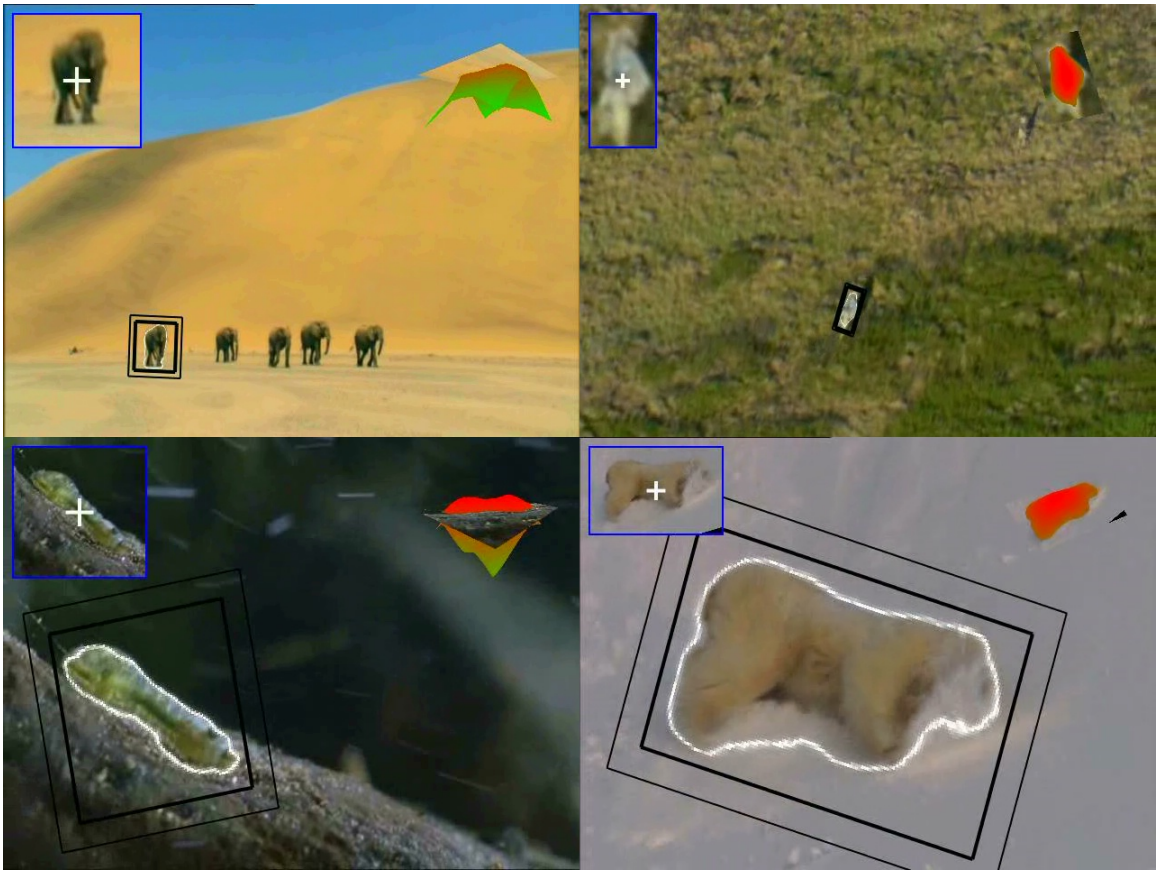


Figure 6.13: Tracking animals montage.

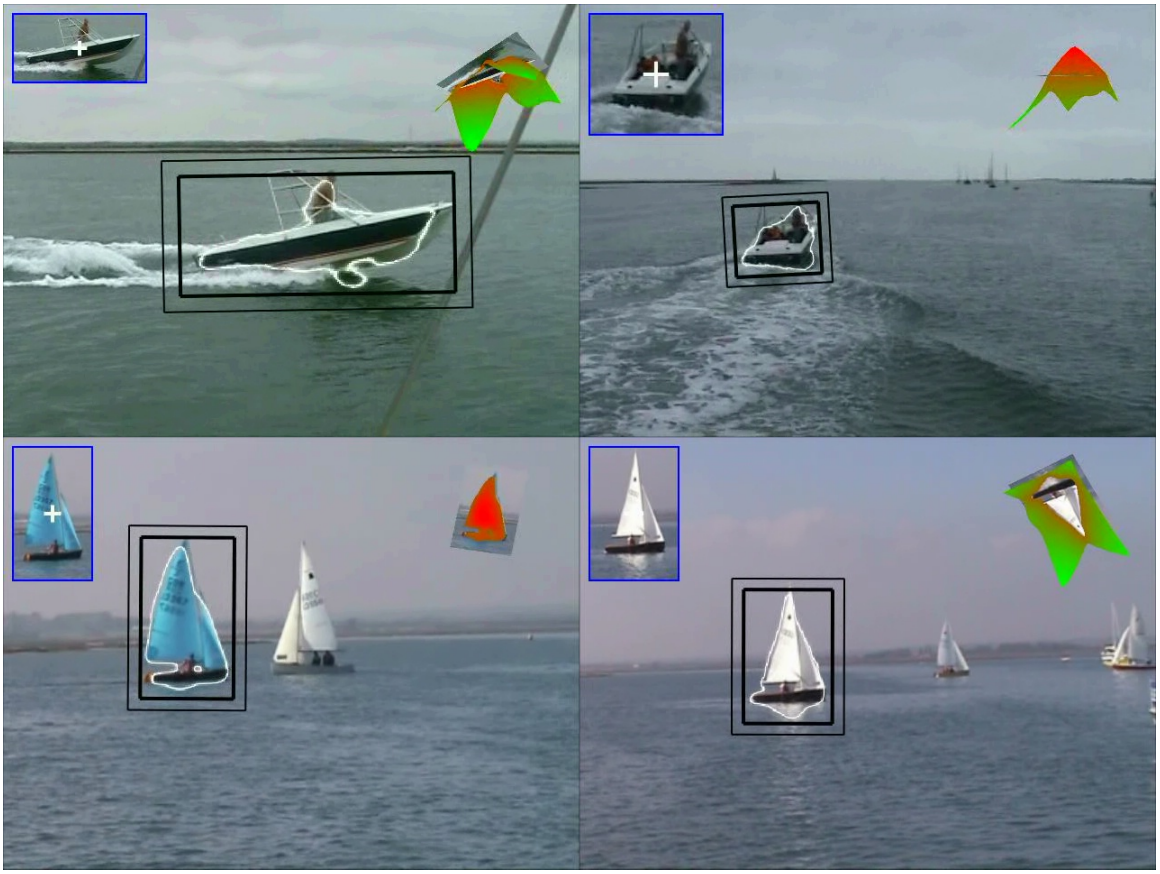


Figure 6.14: Tracking boats montage.

Chapter 7

Tracking Multiple Interacting/Occluding Objects using Pixel-Wise Posteriors

7.1 Introduction

While the previous chapter dealt with tracking a single object, this chapter will introduce a generalised model that allows us to track *multiple objects* in real-time. One approach that has demonstrated significant success in representing the presence of multiple objects, in an image or video, is that of a layered or 2.5D representation [Nitzberg & Mumford, 1990; Wang & Adelson, 1993; Jepson *et al.*, 2002; Tao *et al.*, 2000; Jojic & Frey, 2001; Reid & Connor, 2005]. The image formation process is modelled by multiple layers at different depths, with the background layer being furthest away and then a number of different object layers organised according to their relative depths in the scene. This representation is able to directly model the occlusion process, assuming that objects do not interlock with each other.

This chapter presents a fully probabilistic, generative model for the image formation process, which captures the essence of a layered representation and adds the power, speed and resilience of pixel-wise posteriors. Following the previous chapter, we use implicit contours (level-sets) to represent the boundaries of the objects being tracked [Osher & Paragios, 2003; Paragios & Deriche, 2000; Goldenberg *et al.*, 2001; Chan & Vese, 2001; Cremers, 2006] and pixel-wise posteriors to give better behaved objective functions and hence resilience to noise. The result is, to our knowledge, the first

system that can simultaneously estimate the position, scale, rotation (and potentially other parameters), depth-ordering, figure-figure and figure-ground segmentation for up to twelve occluding/interacting objects, in *real-time*, using standard PC hardware.

It is not obvious from the previous chapter how multiple occluding objects should be handled and so the key contribution of this chapter is: solving the non-trivial task of tracking multiple occluding objects using pixel-wise posteriors. In particular: (i) we begin with a more sophisticated generative model, which directly models the process of inter-object occlusion; (ii) we take the Bayesian approach and marginalise out all nuisance parameters when inferring the best configuration over the multi-object space; (iii) we show how to introduce motion models; (iv) we compute a posterior over the depth-ordering of multiple objects and (v) we demonstrate the ability to track through complete occlusions in challenging situations.

Given our generative model, we are able to perform inference and an optimisation at each frame to find the MAP estimate for the configuration of the multiple objects. An alternative to this approach, which has been used successfully for multi-object tracking, is the use of a non-parametric representation, in particular, the various methods based on particle filtering [Isard & MacCormick, 2001; Vermaak & Doucet, 2003; Khan *et al.*, 2004]. A significant advantage of these methods is that you only need to sample a particle distribution and weight each particle by the likelihood function, which in practice is often easier to implement than a direct optimisation. The downsides are that often a very large number of particles is required to achieve a good approximation of the true posterior and that given this approximation it is not always obvious how to interpret the particle set.

An important aspect to any multi-object tracking system is probabilistic exclusion, which was first demonstrated in a particle filtering context by [MacCormick & Blake, 1999]. Probabilistic exclusion enforces that a single measurement should not be allowed to explain multiple objects, which in practice ensures that several objects do not incorrectly get assigned to the same data. The generative model we propose enforces this directly by only allowing a pixel to be generated from a single object.

Our method only needs an initial bounding box to start tracking an object and therefore has to estimate everything about the objects and their interactions from the incoming video data. In contrast, if 3D models for the objects are available before tracking commences, then it is possible to estimate the full 3D poses of the objects and hence compute occlusion/visibility directly from their 3D structure [Drummond & Cipolla, 2000; Schmaltz *et al.*, 2007]. This can offer superior performance and

accuracy; however, there are many real-world scenarios where this information is not available before tracking commences and so a method such as ours is required that can be initialised without prior knowledge of 3D structure.

The remainder of this chapter is organised as follows: Section 7.2 describes the representation of the objects being tracked; Section 7.3 describes our generative models and their corresponding probability distributions; Section 7.4 outlines our method for tracking; Section 7.5 shows our results and Section 7.6 concludes with a summary and discussion.

7.2 The Representation

Figure 7.1 illustrates how we represent each of the K objects being tracked by their shape $\mathbf{C}_j$, their location in the image $\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)$ and two underlying appearance models: one for the foreground $P(\mathbf{y}|M = m_j^f)$ and one for the background $P(\mathbf{y}|M = m_j^b)$.

Shape: This is represented by the zero level-sets $\mathbf{C}_j = \{\mathbf{x}|\mathbf{\Phi}_j(\mathbf{x}) = 0\}$ of the embedding functions $\mathbf{\Phi}_j(\mathbf{x})$ [Osher & Paragios, 2003; Cremers, 2006]. This is identical to the approach in the previous chapter except there is now a level-set for each object being tracked.

Location: This is described by a warp $\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)$ that takes a pixel location $\mathbf{x}_n$ in object j's coordinate frame and warps it into the image frame according to parameters $\mathbf{p}_j$. Again this is identical to the approach in the previous chapter except there is now a set of pose parameters for each object being tracked.

Appearance models: Each object j has a pair of colour models $P(\mathbf{y}|M = m_j^f)$ and $P(\mathbf{y}|M = m_j^b)$, one for its foreground pixels and one for the nearby background pixels. We use a local background model for each object being tracked, as opposed to a single model shared between objects. We have tried both approaches and we have found that a local background model per object is crucial to the success of the algorithm. This is because the background can vary significantly over the image and successful tracking requires that each object is segmented from the local background i.e. nearby background pixels.

Regions: This is the most significant difference from the previous chapter. The area in each object's coordinate frame is broken into multiple regions R. For a single object there would simply be two regions: foreground and background. Extending

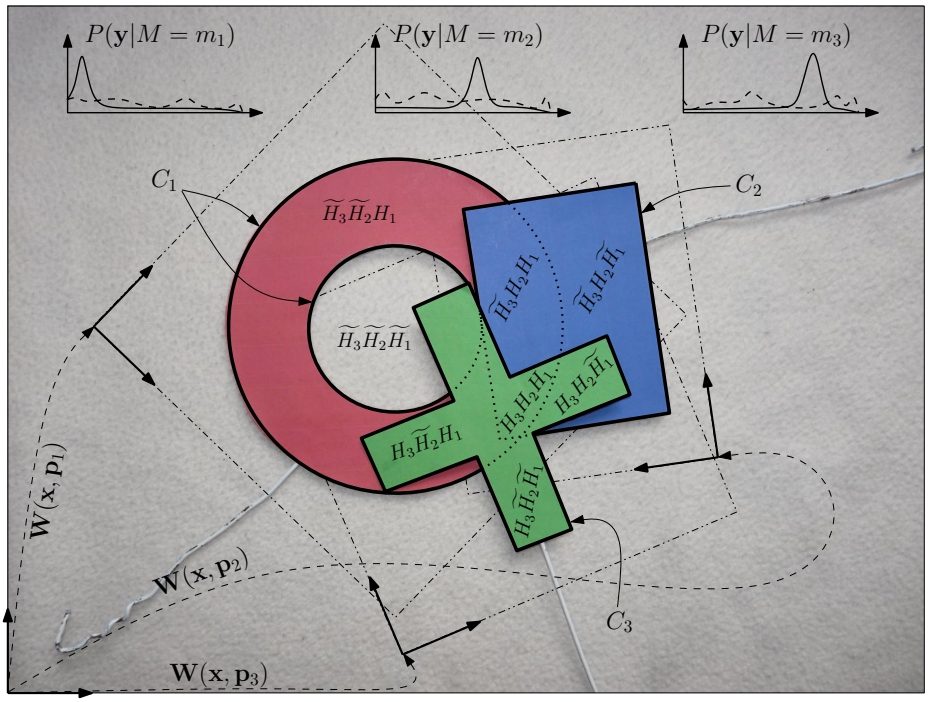


Figure 7.1: Representation for three objects showing: the three pairs of colour models $P(\mathbf{y}|M = m_j)$ where $j = \{1, 2, 3\}$; the three warps into the objects' coordinate frames $\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)$; the three contours $\mathbf{C}_j$ corresponding to the shapes of the three objects and the eight potential regions: $\widetilde{H_3}\widetilde{H_2}\widetilde{H_1}$, $\widetilde{H_3}\widetilde{H_2}H_1$, $\widetilde{H_3}H_2\widetilde{H_1}$, $\widetilde{H_3}H_2H_1$, $H_3\widetilde{H_2}\widetilde{H_1}$, $H_3\widetilde{H_2}H_1$, $H_3H_2\widetilde{H_1}$ and $H_3H_2H_1$, which correspond to different types of overlap for example $\widetilde{H_3}\widetilde{H_2}\widetilde{H_1}$ is the background region and $H_3\widetilde{H_2}H_1$ is the region where object 1 and object 3 overlap in the image.

to two objects there are four regions: background, object 1, object 2 and the overlap between object 1 and 2. In general, for K objects there are 2^K regions required to cover all possible types of interaction (occlusion). Figure 7.1 shows an example of three interacting objects with their eight potential regions. The notation H_j and $\widetilde{H_j}$ can be interpreted as: foreground of object j and not foreground of object j respectively, for example $H_3H_2H_1$ is the foreground of objects 1,2 and 3, in other words the region where these three objects overlap. In summary, we use the following notation:

- N: The number of pixels.
- K: The number of objects.
- $\mathbf{x} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$: The set of pixel locations in the object's coordinate frame.

- $\mathbf{y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_N\}$: The set of pixel values (in our experiments this is a RGB value).
- $\mathbf{I} = \{\{\mathbf{x}_1, \mathbf{y}_1\}, \ldots, \{\mathbf{x}_N, \mathbf{y}_N\}\}$: Image within the objects' coordinate frames.
- $j = \{1, \ldots, K\}$: Object index where K is the number of objects being tracked.
- $\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)$: Warp with parameters $\mathbf{p}_j$ corresponding to object j.
- $M_n = \{m_1^{\{b,f\}}, \ldots, m_K^{\{b,f\}}\}$: Model parameter either background or foreground for one of the K objects, there are $2K$ models.
- D: The discrete depth-ordering of the foreground objects, there are $K!$ possible orderings.
- R_n: The region that the pixel $\mathbf{x}_n$ has been generated from. With K objects there are 2^K possible regions.
- $\mathbf{C}_j$: The contour that segments the foreground object j from the background.
- $\mathbf{\Phi}, \mathbf{p} = \{\{\mathbf{\Phi}_1, \mathbf{p}_1\}, \ldots, \{\mathbf{\Phi}_K, \mathbf{p}_K\}\}$: Shape kernels (level-set embedding functions) and pose parameters.
- $H_\epsilon(z)$: Smoothed Heaviside step function, where ϵ is the smoothing parameter.
- $\delta_\epsilon(z)$: Smoothed Dirac delta function, where ϵ is the smoothing parameter.
- $H_j = H_\epsilon(\mathbf{\Phi}_j(\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)))$: Shorthand for the smoothed Heaviside step function applied to object j's shape kernel.
- $\widetilde{H_j} = 1 - H_\epsilon(\mathbf{\Phi}_j(\mathbf{W}(\mathbf{x}_n, \mathbf{p}_j)))$: Shorthand for one minus the smoothed Heaviside step function applied to object j's shape kernel.

7.3 The Generative Models

We will now consider two different approaches for modeling the relative depth-ordering of the objects. The first assumes that *each pixel* belonging to an object carries its own relative depth. This is a very general model and allows objects to interact in complicated ways, for example meshing hands. The second approach is more restrictive and assumes that *each frame* carries a relative depth. This is more in keeping

with the traditional layered representations and means that the pixels belonging to an object all share a common relative depth.

Figure 7.2 shows generative models that correspond to the two approaches and have the following variables: $\mathbf{\Phi}$ is the shape kernels, one per object; $\mathbf{p}$ is the set of parameters describing the rigid transformations from each object's coordinate frame to the image frame; D is the discrete depth-ordering of the objects (there are $K!$ orderings); M_n is the appearance model (there are $2K$ appearance models); R_n is the region the pixel is generated from (there are 2^K regions); $\mathbf{y}_n$ is the pixel colour and $\mathbf{x}_n$ is the pixel location. The difference between the two models is that the one on the left includes D_n within the plate, meaning that depth is treated as a local (pixel-wise) parameter, whereas the other model treats D as a global (frame-wise) parameter.

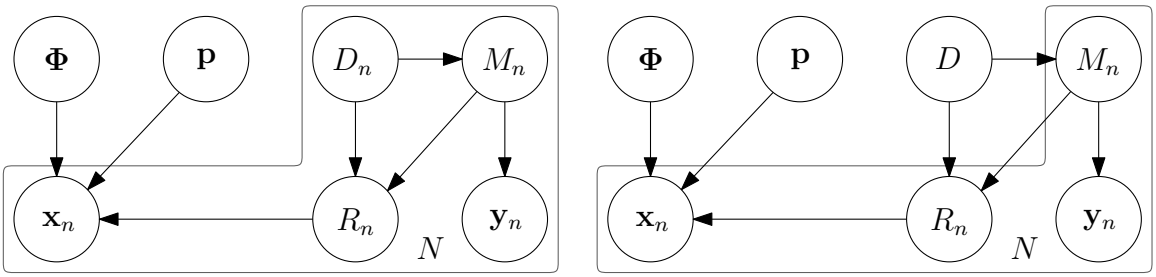


Figure 7.2: Generative models representing the image as a bag-of-pixels: (left) the discrete depth-ordering is a local parameter and (right) the discrete depth-ordering is a global parameter (note:- the gray polygon denotes a plate over the N pixels in the objects' coordinate frames).

The intuition behind these graphical models is that given the shape $\mathbf{\Phi}$, the pose $\mathbf{p}$ and the depth-ordering D, you can first sample a particular appearance model M_n, then sample a region R_n where the appearance model is present and finally sample a $\{\mathbf{x}_n, \mathbf{y}_n\}$ pixel pair, which tells you where the pixel is $\mathbf{x}_n$ and what colour it has $\mathbf{y}_n$. If you were to sample from this generative model N times, then you would end up with an image that is representative of the original apart from the fact that pixels within an object's contour are jumbled up, which is because we are modeling the colour distributions $P(\mathbf{y}_n|M_n)$ rather than the specific spatial arrangement of pixels. The joint distribution for a single pixel given the left-hand generative model is:

$$P(\mathbf{x}_n, \mathbf{y}_n, \mathbf{\Phi}, \mathbf{p}, R_n, M_n, D_n) = P(\mathbf{\Phi})P(\mathbf{p})\times$$
$$P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)P(R_n|D_n, M_n)\times$$
$$P(\mathbf{y}_n|M_n)P(M_n|D_n)P(D_n). \quad (7.1)$$

By performing inference we can obtain the MAP estimate of $P(\mathbf{\Phi}, \mathbf{p}, D_n|\mathbf{I})$ and compute the best shape, pose and discrete depth-ordering to explain the observed image data. The way that we achieve this is to first condition on $\mathbf{x}_n$ and $\mathbf{y}_n$, where $P(\mathbf{x}_n)$ is constant and

$$P(\mathbf{y}_n) = \sum_{M_n, D_n} P(\mathbf{y}_n|M_n)P(M_n|D_n)P(D_n). \quad (7.2)$$

Then second, marginalise over R_n and M_n to give us the pixel-wise posterior:

$$P(\mathbf{\Phi}, \mathbf{p}, D_n|\mathbf{x}_n, \mathbf{y}_n) = \frac{1}{P(\mathbf{x}_n)P(\mathbf{y}_n)}P(\mathbf{\Phi})P(\mathbf{p})\times$$
$$\sum_{R_n, M_n} P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)P(R_n|D_n, M_n)\times$$
$$P(\mathbf{y}_n|M_n)P(M_n|D_n)P(D_n). \quad (7.3)$$

Finally, we take the product over the pixel sites to get the posterior given the image $\mathbf{I}$:

$$P(\mathbf{\Phi}, \mathbf{p}, D|\mathbf{I}) = P(\mathbf{\Phi})P(\mathbf{p})\prod_{n=1}^{N}\frac{1}{P(\mathbf{y}_n)}\times$$
$$\sum_{R_n, M_n} P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)P(R_n|D_n, M_n)\times$$
$$P(\mathbf{y}_n|M_n)P(M_n|D_n)P(D_n). \quad (7.4)$$

The corresponding posterior taking D as a global parameter is:

$$P(\mathbf{\Phi}, \mathbf{p}, D|\mathbf{I}) = P(\mathbf{\Phi})P(\mathbf{p})P(D)\prod_{n=1}^{N}\frac{1}{P(\mathbf{y}_n)}\times \sum_{R_n, M_n} P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)P(R_n|D, M_n)\times P(\mathbf{y}_n|M_n)P(M_n|D), \quad (7.5)$$

where there is now a single D for all pixels in the current frame (see the right-hand generative model in Figure 7.2).

The Probability Distributions

We will now explain each of the distribution terms which comprise (7.4) and (7.5) in detail:

- $P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)$: is the probability of the pixel location $\mathbf{x}_n$ given the shape $\mathbf{\Phi}$, the pose $\mathbf{p}$ and the region R_n, which can be written in terms of the blurred Heaviside step function H_j and one minus the blurred Heaviside step function $\widetilde{H_j}$. For K objects there are 2^K regions, so for example with 3 objects there are 8 regions, which take the form:

$$\begin{aligned} P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n = 0) &= \widetilde{H_3}\widetilde{H_2}\widetilde{H_1}/\eta_0 \\ P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n = 1) &= \widetilde{H_3}\widetilde{H_2}H_1/\eta_1 \\ P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n = 2) &= \widetilde{H_3}H_2\widetilde{H_1}/\eta_2 \\ &\cdots \\ P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n = 7) &= H_3H_2H_1/\eta_7, \end{aligned} \quad (7.6)$$

where the terms on the right-hand side follow a binary sequence (this is related to the work on multi-phase level-sets [Vese & Chan, 2002]) and are normalised by η_r so that the distributions sum to one.

- $P(R_n|D_n, M_n)$: is the probability of a region R_n given a discrete depth-ordering D_n and a model M_n. This is computed using a ratio of region areas, an example using the object arrangement in Figure 7.1 would be:

$$P(R_n = 7|D_n, M_n = m_3^f) = \eta_7/(\eta_4 + \eta_5 + \eta_6 + \eta_7). \tag{7.7}$$

- $P(\mathbf{y}_n|M_n)$: is the probability of the colour $\mathbf{y}_n$ given the model M_n (represented using a colour histogram).

- $P(M_n|D_n)$: is the probability of a model M_n given a discrete depth-ordering D_n. This is computed using a ratio of region areas, an example using the object arrangement (depth-ordering) in Figure 7.1 would be:

$$P(M_n = m_3^f|D_n) = (\eta_4 + \eta_5 + \eta_6 + \eta_7)/\eta, \tag{7.8}$$

where η is the sum of η_r over all regions, which equals the number of pixels N.

- $P(D_n)$ and $P(D)$: are the prior distributions on the discrete depth-orderings. They are taken to be the uninformative uniform distribution.

- $P(\mathbf{\Phi})$: is the prior on the shape $\mathbf{\Phi}$, which we take to be:

$$P(\mathbf{\Phi}) = \prod_{n=1}^{N} \frac{1}{\sigma\sqrt{2\pi}} \exp - \frac{(|\nabla \mathbf{\Phi}(\mathbf{x}_n)| - 1)^2}{2\sigma^2}, \tag{7.9}$$

where σ specifies the relative weight of the prior. This is identical to the prior term used in the previous chapter and automatically maintains approximate signed distance functions for the level-set embedding functions.

- $P(\mathbf{p})$: is the prior on the pose $\mathbf{p}$, which is either taken to be the uninformative uniform distribution or a motion model $P(\mathbf{p}^t|\mathbf{p}^{t-1})$, which will be described in Section 7.4.

Note:- It should be pointed out that when substituting these distributions into Equations (7.4) and (7.5) several expressions cancel, simplifying the final implementation (exact details skipped for brevity).

7.4 Tracking

Given the posterior distributions (7.4) and (7.5) our objective is to compute the MAP estimate for the pose $\mathbf{p}$, the shape $\mathbf{\Phi}$ and the discrete depth-ordering D. Ideally these would be maximised jointly, but the high dimensionality of the joint space makes this prohibitively expensive and so as an approximation we break the problem into five steps: (i) a rigid registration to account for any rigid motion between frames; (ii) a segmentation to account for any local shape deformation; (iii) a posterior over depth-ordering, holding the pose and the shape constant; (iv) updating the appearance models and (v) drift correction. We will now explain each of these five steps in greater detail.

Rigid Registration and Motion Modeling

If we begin by taking $P(\mathbf{p}^t|\mathbf{p}^{t-1})$ to be the uninformative uniform distribution (i.e. no motion model), then the MAP estimate for $\mathbf{p}$:

$$\mathbf{p} = \arg\max_{\mathbf{p}} \left\{ \sum_{n=1}^{N} \log P(\mathbf{\Phi}, \mathbf{p}, D|\mathbf{x}_n, \mathbf{y}_n) \right\} \tag{7.10}$$

can be solved (similar to Section 6.5) with an approximate version of the second-order Newton optimisation scheme:

$$\Delta\mathbf{p} = \left[\sum_{n=1}^{N} B_n^2 \mathbf{J}_n^T \mathbf{J}_n \right]^{-1} \sum_{n=1}^{N} \mathbf{J}_n^T B_n, \tag{7.11}$$

where

$$\mathbf{J}_n = \frac{\partial H_\epsilon}{\partial \mathbf{\Phi}} \frac{\partial \mathbf{\Phi}}{\partial \mathbf{W}} \frac{\partial \mathbf{W}}{\partial \mathbf{p}} = \delta_\epsilon(\mathbf{\Phi}) \nabla \mathbf{\Phi}(\mathbf{x}_n) \frac{\partial \mathbf{W}}{\partial \mathbf{p}} \tag{7.12}$$

and B_n is scalar value, which is obtained by differentiating (7.3) with respect to $\mathbf{p}$. Although the exact derivation of B_n is skipped for brevity it should be pointed

out that the differentiation of (7.3) leads to a sum of positive and negative weights, where the weights are related to the likelihoods $P(\mathbf{y}_n|M_n)$ and the region probabilities $P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n)$.

Let us now introduce a constant velocity motion model $P(\mathbf{p}^t|\mathbf{p}^{t-1})$ with Gaussian noise, which will add an extra term to (7.10) and modify (7.11) to include prior terms:

$$\Delta\mathbf{p} = \left[\sum_{n=1}^{N} B_n^2 \mathbf{J}_n^T \mathbf{J}_n + \mathbf{J}_p^T \mathbf{J}_p\right]^{-1} (\sum_{n=1}^{N} \mathbf{J}_n^T B_n + \mathbf{J}_p^T \mathbf{B}_p), \tag{7.13}$$

where $\mathbf{J}_p$ represents the Jacobian due to the prior information and $\mathbf{B}_p$ is the error vector for the prior. This equation has the property that as an object becomes gradually occluded, the B_n terms will reduce, increasing the relative weight of the prior. At the point an object becomes completely occluded the B_n terms will be zero and (7.13) will reduce to a motion model prediction equation. Note:- The terms $\mathbf{J}_n$ and $\mathbf{J}_p$ remain constant with respect to the shape and can therefore be precomputed for efficiency during the registration step (refer to [Baker & Matthews, 2004] for details).

Depth-Ordering

We have two methods of dealing with the depth parameter depending on whether we treat it as a local (pixel-wise) or a global (frame-wise) parameter. Treating it as a local parameter means we can maximise for D_n within the registration step (7.10) by performing a pixel-wise maximisation over D_n:

$$\mathbf{p} = \arg\max_{\mathbf{p}} \left\{ \sum_{n=1}^{N} \max_{D_n} \{\log P(\mathbf{\Phi}, \mathbf{p}, D_n|\mathbf{x}_n, \mathbf{y}_n)\} \right\}. \tag{7.14}$$

This method picks the best D_n out of the $K!$ choices at each pixel and therefore has no consistency at the frame level. The benefit of this approach is that it can deal with objects interacting in complicated ways, for example: a pair of hands meshing fingers. The downside of this approach is that it is often easier to choose an incorrect depth-ordering than it is to explain the image data correctly, which results in increased

sensitivity to noise and a tendency to assume interacting pixels are occluded, when they are not. The alternative to this is to treat D as a global parameter and assume that objects do not interlock (we show in Section 7.5 that this increases resilience to noise). We first compute the following posterior over the $K!$ depth-orderings:

$$P(D|\mathbf{\Phi}, \mathbf{p}, \mathbf{I}) = P(D) \prod_{n=1}^{N} \frac{1}{P(\mathbf{y}_n)} \sum_{R_n, M_n} P(\mathbf{x}_n|\mathbf{\Phi}, \mathbf{p}, R_n) P(R_n|D, M_n) P(\mathbf{y}_n|M_n) P(M_n|D) \quad (7.15)$$

and then use the MAP estimate for D when optimising for the pose $\mathbf{p}$ and shape $\mathbf{\Phi}$ in the next frame.

Grouping

An efficient implementation must address the problem of exponential growth in $K!$. We deal with this by forming groups of objects, so that the amount of inter-object overlap within a group is maximised, subject to a maximum group size of three. The depth posterior is then computed on a per group basis. The result is that depth posteriors are only calculated where they are most necessary. Figure 7.9 shows a typical example of how objects are grouped, with the white lines representing the object groups. By limiting the maximum group size to three, we are able to track twelve or more objects comfortably in real-time.

Segmentation, Appearance Learning and Drift Correction

The methods for segmentation, appearance learning and computing drift correction are the same as the approach taken in the previous chapter, except that the level-set evolution in the segmentation step and appearance learning are modulated by a parameter λ_n, which effectively turns off learning in areas where objects overlap. The parameter λ_n is defined as the probability that objects do not overlap at a given pixel, which is computed directly using the $P(R_n|\mathbf{\Phi}, \mathbf{p}, \mathbf{x}_n)$ terms.

7.5 Results

We will first quantitatively evaluate the performance of the local vs. global depth parameter as described in Section 7.4. To test their performances against each other, we have generated a simple simulation containing three objects that repeatedly cross each other in different ways, we then add varying amounts of Gaussian noise to the pixel colours (see Figure 7.3). Given that we have the ground truth for the sequence, we are able to compute the normalised pose error $\sum_{i=1}^{F}\sum_{j=1}^{K}(\mathbf{p}_j^i - \hat{\mathbf{p}}_j^i)^T\mathbf{R}^{-1}(\mathbf{p}_j^i - \hat{\mathbf{p}}_j^i)/K/F$ i.e. a normalised error between the estimated pose and the true pose, where $\mathbf{R}^{-1} = diag[1, 1, 100, 180/\pi]$ and F is the number of frames[1]. Figure 7.4 demonstrates that superior performance is achieved in the presence of noise using the global parameter. This is no surprise given that the global parameter is a simpler model than the local parameter, yet both can explain the data exactly. It is also worth noting that using a motion model (MM) confers a measurable improvement in performance.

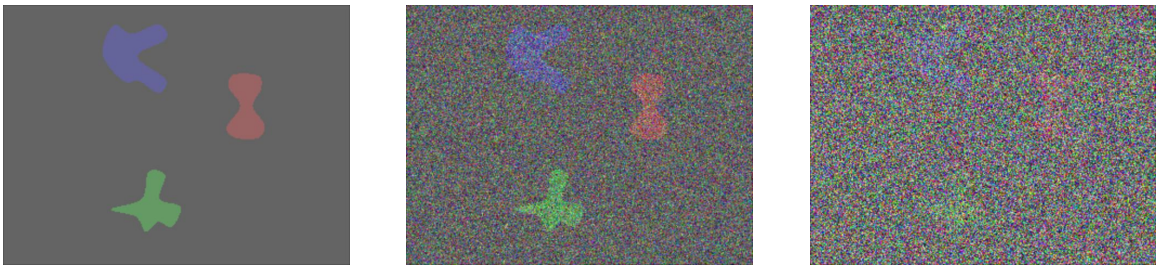


Figure 7.3: The simulation: these objects move around (translation, scale and rotation), overlapping each other in different ways. This shows varying amounts of noise, corresponding to standard deviations of 0, 40, and 70 pixels (from left to right).

Figure 7.5 illustrates that marginalising out nuisance variables to obtain the pixel-wise posterior (7.3) achieves superior resilience to noise compared with optimising the joint (7.1) directly. This graph was generated using a similar simulation to Figure 7.3 but with only a single object. The difference in performance is significant.

We will now qualitatively look at three sequences, the first is 1600 frames long and consists of three distinctive shapes that are translating, rotating and scaling, as well as occluding each other in different ways. Figure 7.6 highlights a small section of this sequence where all three objects interact. In particular, the green object passes behind the blue object (frame 1147), which then passes behind the red object (frame 1151), resulting in the green object being completely occluded and the blue object

[1]The purpose of $\mathbf{R}^{-1}$ is to normalise the pose dimensions so that an angle measured in radians can be compared against a scaling factor or a translation measured in pixels.

only being visible through the hole in the red object (frame 1157). Finally, by the end of the sequence (frame 1172) all objects have been successfully tracked through the occlusion.

The second sequence is 800 frames long and shows a typical security video taken in a shopping centre [Fisher, 2004] (see Figure 7.7). Our method successfully tracks four people as they walk from one end of the corridor to the other, occluding each other in different ways and crossing over completely between frames 0 and 207. In frame 369, a man in a striped shirt can be seen approaching from the right, then in frame 401 (zoomed in) he crosses behind the people we are tracking. At this point the green object shrinks to accommodate the unmodelled occlusion event, if we had been tracking the person in the striped top this would not have been the case, because the occlusion would have been modelled within the system.

Finally, the third sequence is 2000 frames long and shows a person carrying out actions that may be similar to those required for a human machine interface, see Figure 7.8. We successfully track both hands and the face for the duration of the sequence. As an example application, the tracker has been used to anonomise the face using the estimate for the shape and the pose. Frame 609 shows both hands occluding the face, sweeping from top to bottom. This is the sort of behaviour that would often break a tracking system. Frames 1231 to 1258 show the person rolling their hands one in front of the other, again behaviour that would often break a tracking system.

The depth-ordering is shown in the results by drawing the objects according to the MAP depth-order, which means that the coloured contour for an object will be hidden if it is behind another object. Generally the system gets the depth-ordering correct,

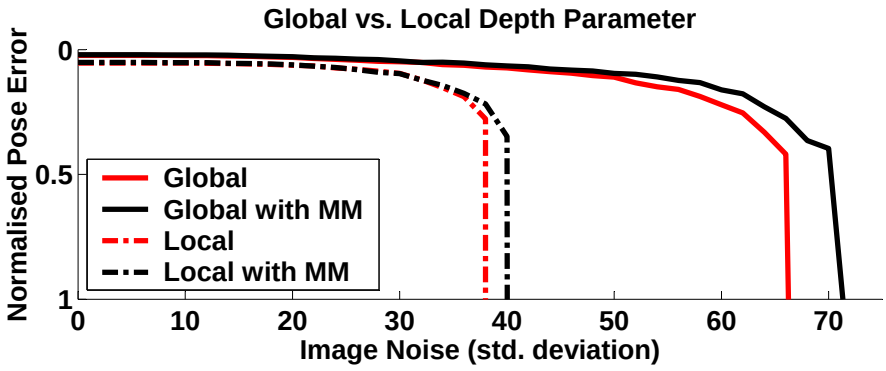


Figure 7.4: Comparison of global vs. local depth parameters, with and without a motion model (MM). The vertical regions in the plots at standard deviations of 40 and 70 correspond to catastrophic failures in tracking.

but occasionally makes mistakes if the appearance of the objects being tracked are similar. For an example consider frame 912 in Figure 7.8, where the person's hands overlap and the system makes a mistake about the correct depth-ordering. Although these mistakes are sometimes made, it is not necessarily detrimental to tracking because if the objects have similar appearances then the incorrectly classified pixels will have a weak influence during the registration stage.

Figures 7.10-7.12 show a selection of applications of the multi-object tracker to real-world examples. Figure 7.10 shows an application of the tracker to a marine environment where we track a selection of dinghys in a race and their safety boat. Figure 7.11 shows the tracker being used to track ice hockey players. Figure 7.12 shows the tracker being used to track every football player on a pitch, which is impressive for the following reasons: (i) no prior on human shape or appearance is used; (ii) no geometric information regarding the pitch is used and (iii) a detector is only used to initiate new tracks and not to correct tracking mistakes. Throughout the sequence the frame rate is always greater than 25 frames per second and averages 40 frames per second. Each of these figures display a stabilised video of the objects being tracked. These are shown as small tiles, which are overlaid on the video with a blue border.

7.6 Conclusions

We have presented a tracking method that represents multiple interacting/occluding objects with a probabilistic generative model. This model includes the discrete depth-ordering for the objects, their locations in the image represented by rigid transformations and their shapes represented with implicit contours. We show how to efficiently

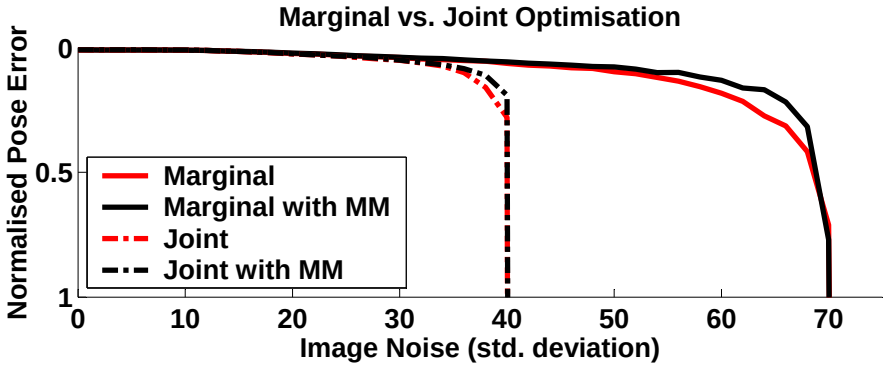


Figure 7.5: Comparison of optimising the marginal (7.5) vs. the joint distribution (7.1), with and without a motion model (MM).

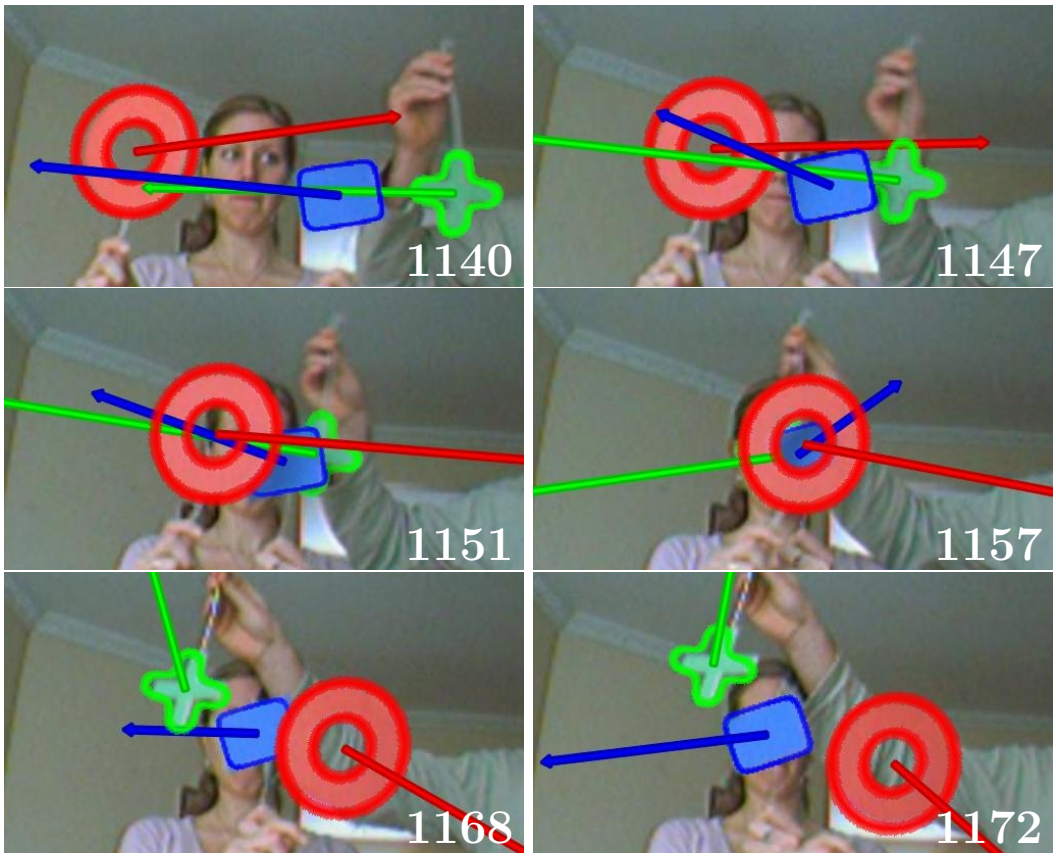


Figure 7.6: Tracking three distinctive shapes. The frame number is in the bottom right, the coloured contours show the objects' pose and shape estimates and the coloured arrows show the objects' estimated velocities.

perform inference on this generative model and compute the MAP estimate for the poses, shapes and discrete depth-ordering of the objects. By using implicit contours and a novel second-order registration scheme, we are able to compute this inference in real-time i.e. 30Hz. A key aspect to the success of the system is the fact that we marginalise out all nuisance parameters analytically, leading to pixel-wise posterior terms as opposed to pixel-wise likelihoods. We demonstrate using quantitative results that this provides superior resilience to noise in the image. We have explored two possible methods for representing the discrete depth-ordering, one based on a local (pixel-wise) parameter and the other on a global (frame-wise) parameter. We show quantitatively that the global parameter provides greater performance in difficult conditions. We have also shown how motion models can be included within our second-order registration step and how this enables the system to track complete occlusions. The system has been tested on a variety of challenging video sequences.

This concludes the second part of the dissertation, which has proposed two novel

methods for visual tracking based on pixel-wise posteriors as opposed to pixel-wise likelihoods. These methods can be used to acquire stabilised video streams of objects of interest with minimal prior information about the object itself (just a bounding box). The next part of the dissertation will describe how these methods can be combined with our HSLAMIDE to obtain an information rich estimate of the environment surrounding a marine vessel.

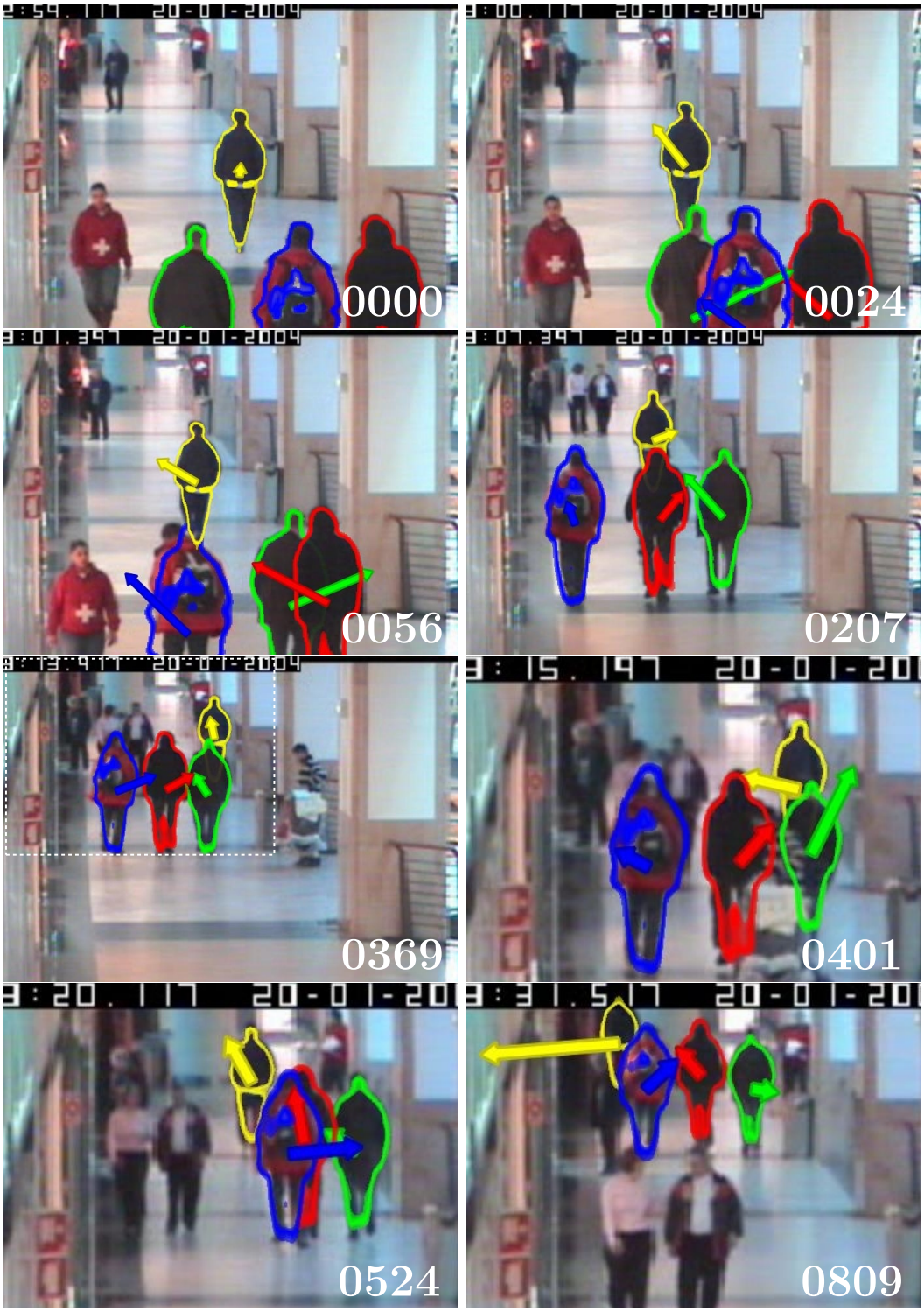


Figure 7.7: Tracking four people in a corridor [Fisher, 2004]. The frame number is in the bottom right, the coloured contours show the objects' pose and shape estimates and the coloured arrows show the objects' estimated velocities. The white dashed box in frame 369 shows the zoomed in area that is used for subsequent frames.

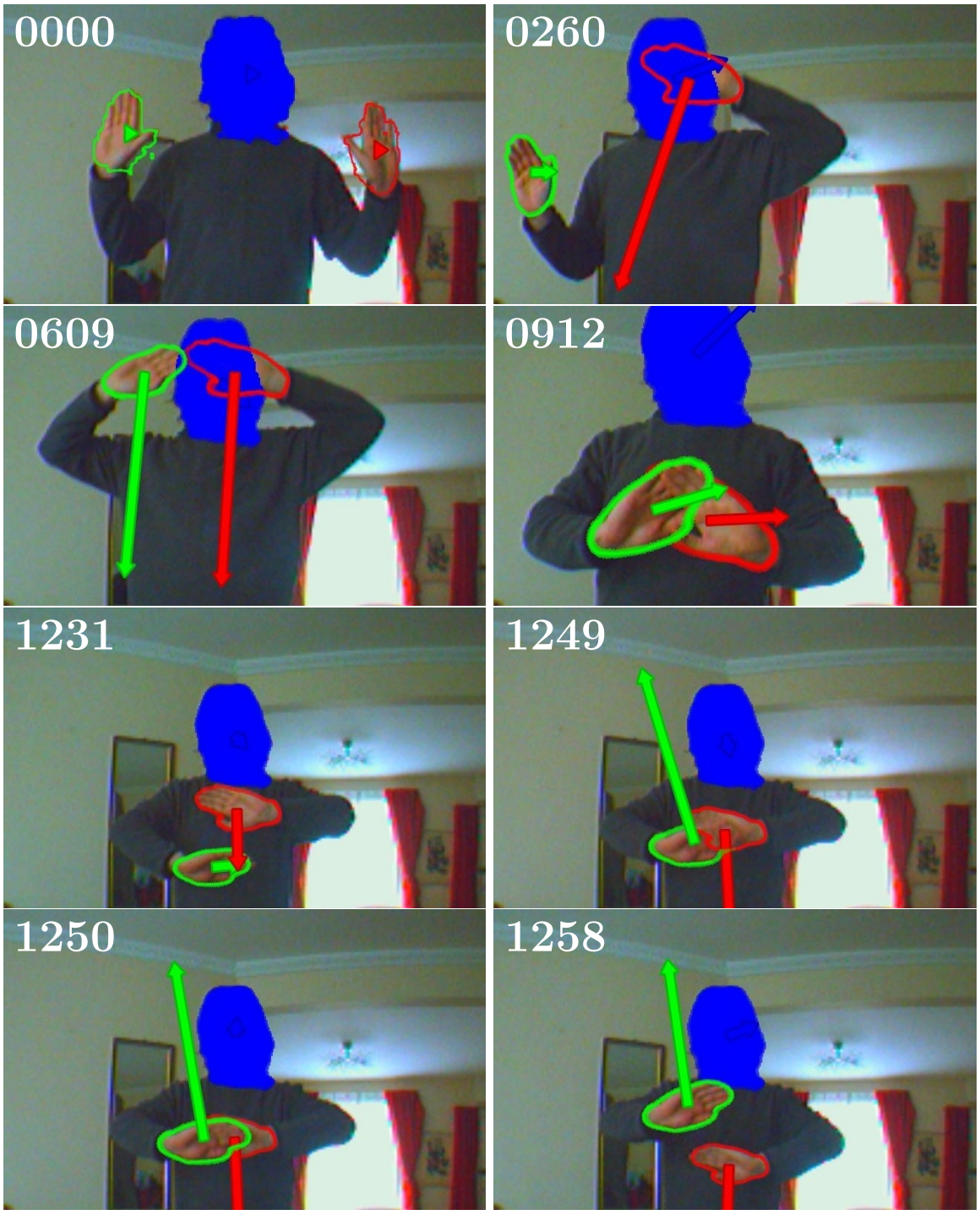


Figure 7.8: Tracking face and hands sequence. The frame number is in the top left, the coloured contours show the objects' pose and shape estimates and the coloured arrows show the objects' estimated velocities. The face has been anonomised using the output from the tracker.

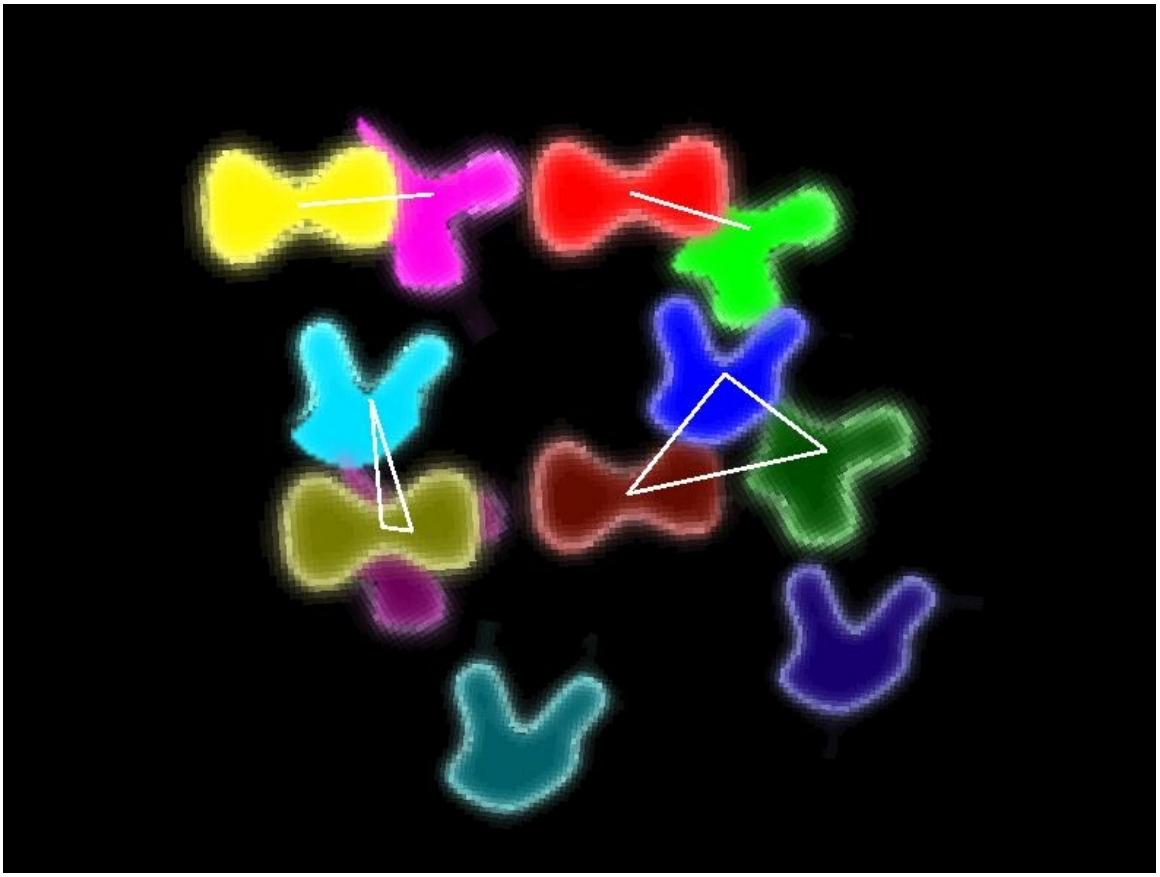


Figure 7.9: Shows the system tracking twelve interacting/occluding objects in real-time. The white lines show the current grouping of objects, which is used to limit the exponential growth when computing depth-orderings.

Figure 7.10: Tracking boats.

Figure 7.11: Tracking ice hockey players.

Figure 7.12: Tracking football players.

Part III

System Integration and Conclusions

The first two parts of this dissertation have shown how we can estimate metric and visual information using marine RADAR and colour imagery. This part of the dissertation will describe how these theoretical contributions can be combined to form an integrated system, which can be used for enhanced situational awareness in marine environments, and finishes with a concluding discussion of the dissertation. The aim of the system integration was to produce a prototype marine surveillance system, which used a mobile vehicle (a boat with RADAR and a Pan-Tilt-Zoom (PTZ) camera) to secure a pre-defined area in the marine environment. This prototype system was demonstrated live to an audience of industry experts, governmental advisors, the navy, special forces, the police and politicians, over a period of two days and was set up to secure the entrance of a river in the Thames estuary.

Chapter 8

Integration

8.1 Introduction

The first part of the dissertation described how SLAM can be extended to handle dynamic objects and how a novel hybrid representation can be used to efficiently represent the marine environment. A key benefit of this approach is that a mobile vehicle can be used to estimate metric information, which has significant advantages for marine applications. Unlike terrestrial surveillance applications where sensors can nearly always be placed at suitable fixed geographical locations, marine surveillance suffers from the limit of the coast-line when placing fixed sensors.

The second part of the dissertation introduced a novel approach for doing robust, real-time visual tracking using pixel-wise posteriors as opposed to the traditional pixel-wise likelihoods. By using a mobile vehicle and visual tracking with PTZ camera(s) the effective visual range for marine surveillance can be extended. This has the significant advantage that visual contact can be made earlier, giving the operators more time to make decisions.

This chapter describes a prototype system that can protect a pre-defined area in the marine environment from potential security threats (see Appendix D for a detailed concept). This system uses the metric methods presented in Part I and the visual methods presented in Part II, to produce a real-time map of the marine environment that contains both metric and visual information. Automatic visual contact is made using the PTZ camera and the acquired visual information is automatically associated with objects in the metric map. This means that an operator is able to quickly cross reference metric information to visual information and vice-versa. The result is that

the system can be used to very quickly establish not only where objects are and what they are doing, but also what they look like. We have demonstrated that by using sensors mounted on a mobile vehicle (Part I of the dissertation) and visual tracking (Part II of the dissertation) it is possible to provide *enhanced situational awareness in marine environments*.

This chapter begins in Section 8.2 with an overview of the prototype system; Section 8.3 describes the design of a high performance PTZ device; Section 8.4 gives a brief overview of the user interface; Section 8.5 takes the reader through an example application of the system and Section 8.6 finishes with conclusions.

8.2 System Overview

The prototype system was distributed over three geographical locations: the sensor platform / mobile vehicle, which was a 10m yacht fitted out with the sensors; the control centre, which was a land based installation that had the main user interface and the mobile security asset, which was a 4.5m Rigid Inflatable Boat (RIB) fitted out with a cut down user interface. Figure 8.1 shows pictures of these locations.

The key benefits of distributing the system over different locations are: (i) the sensors can be placed at the optimum location for the task at hand; (ii) the decision makers can be located in a comfortable shore based control centre and (iii) the operators of the mobile security asset benefit from a reduced workload by shifting all planning and decision making to the shore based control centre.

Figure 8.2 shows how the prototype system is broken into five layers of system components: (i) the sensors; (ii) data acquisition and control; (iii) ship based processing; (iv) a data link and (v) shore based processing. All high level data exchange is done using the User Datagram Protocol (UDP) over Ethernet. We will refer to this process as distributing data or communication. We will now describe each of the layers in more detail.

The Sensors

The system uses five main types of sensor:

- **RADAR:** We use a standard X-band yacht RADAR. This device has an angular

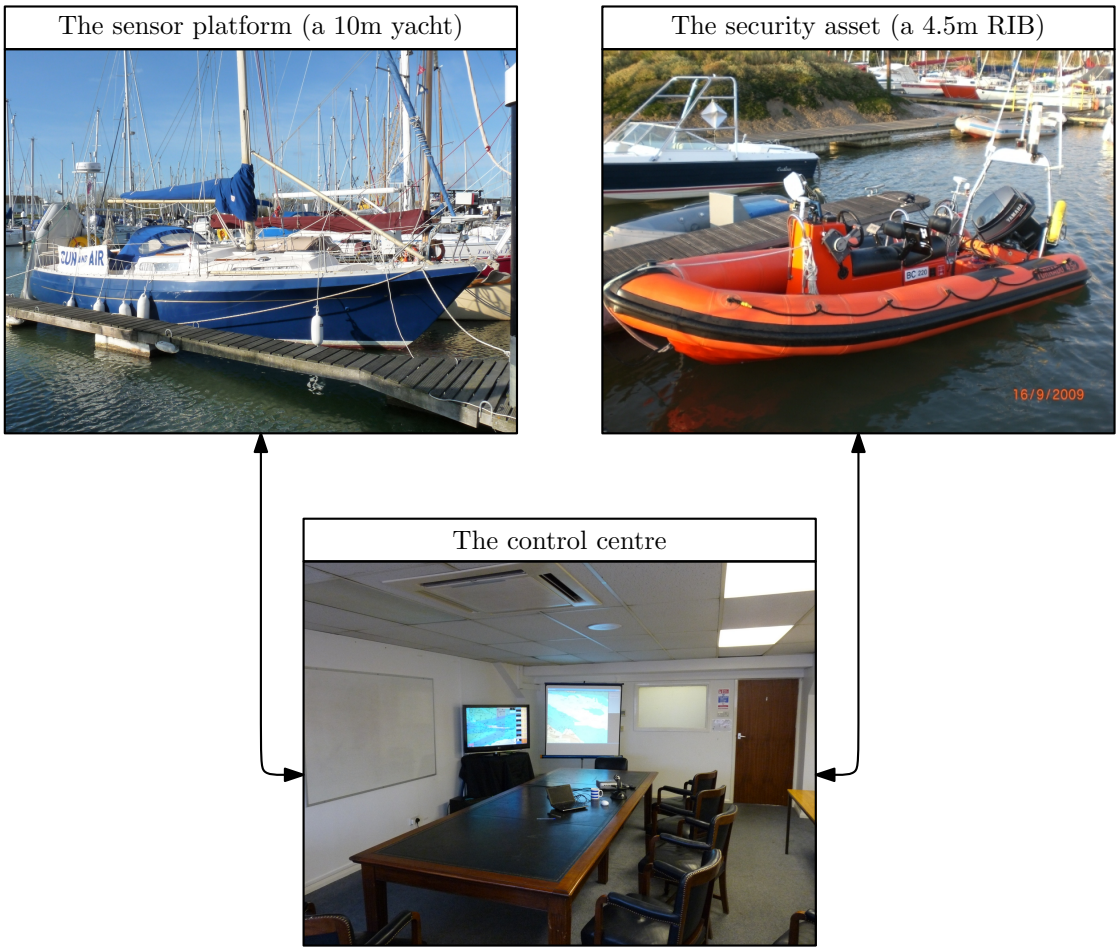


Figure 8.1: Geographical locations.

resolution of 3.6 degrees and has a maximum and minimum range of 48 nautical miles and 20m respectively. We limit the range at 1 nautical mile giving an effective range resolution of approximately 6m. Like all low-end marine RADAR products, the device suffers from significant amounts of clutter (false-positives).

- **PTZ:** This pan-tilt-zoom device was designed, built and tested as part of this DPhil. It is able to rotate 360 degrees and fixate to 1/1000th of a degree within 600ms. These impressive performance characteristics allow us to use the sensor to quickly saccade from one object to another, taking pictures and videos that are then used to enrich the metric information with visual information. This device will be explained in greater detail in Section 8.3.

- **Inertial Measurement Unit (IMU):** This is a miniature attitude and heading reference system. It uses Micro-Electro-Mechanical Systems (MEMS) inertial sensors and magnetometers to estimate the 3DOF attitude and heading

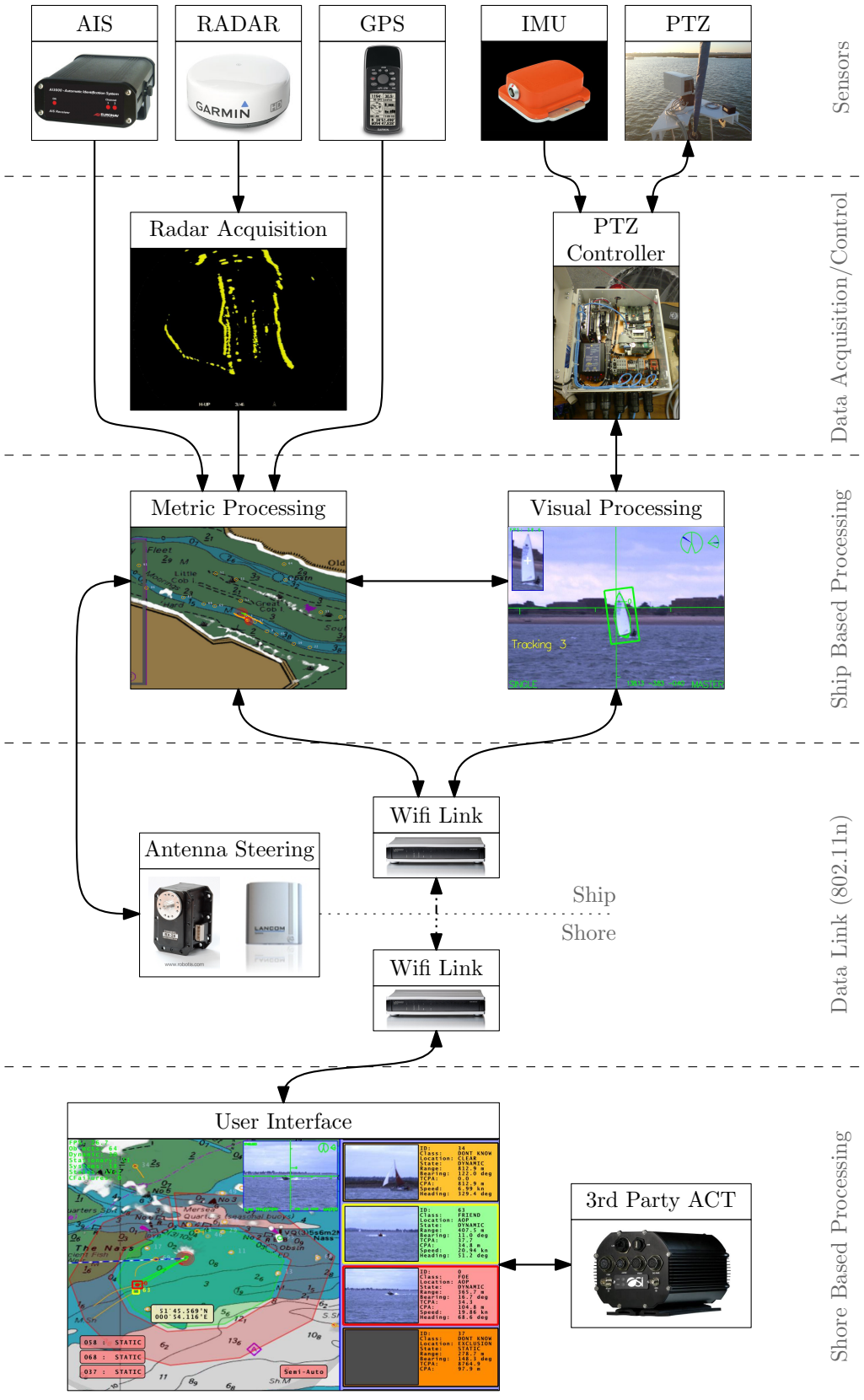


Figure 8.2: System components.

at 100Hz. We use this sensor for two purposes: (i) to stabilise the PTZ and (ii) as a heading reference unit. The sensor has a dynamic performance error of approximately 2 degrees RMS. The heading reference can optionally be fed into the HSLAMIDE system; otherwise, the estimated map is manually aligned with a marine chart.

- **GPS:** This sensor uses satellites to measure its absolute position to within a few meters anywhere on the globe. We have an option of feeding this into the HSLAMIDE system; otherwise, the estimated map is manually aligned with a marine chart.

- **Automatic Identification System (AIS):** This is a marine transponder system that has to be fitted to all ships with a gross tonnage of 300 or more tons and to all commercial passenger vessels. Each transponder broadcasts the vessel's Maritime Mobile Service Identity (MMSI) number along with the position, course, speed and other data. We associate this data with the objects in the hybrid map and use it to automatically tag vessels that we know about.

Data Acquisition and Control

The GPS, AIS and IMU sensors all have serial interfaces, which are brought into processing units, accurately timestamped and then distributed to multiple clients. The raw RADAR video is digitised using a PC and a video digitiser, and then distributed. The PTZ device is controlled by our PTZ controller, which deals with all analogue interfacing, control and inertial stabilisation. The PTZ controller has a simple communication interface to the higher level processes.

Ship Based Processing

The system on the boat runs both SLAM in dynamic environments using hybrid mapping (as presented in Chapters 4 and 5) and visual tracking (as presented in Chapters 6 and 7). The system automatically uses the PTZ camera to make visual contact and build a visual database of objects within sensor range. This visual database is a set of short video sequences that are automatically associated to objects in the metric map. The result is a hybrid map of the environment that contains both metric and visual information. This is hybrid map is then sent back to the shore based installation over

a wireless link, where the operators can interact with the data and control the PTZ camera.

Data Link

The data link was implemented using 802.11n wireless technology and could maintain a throughput of 20Mbps-100Mbps over a distance of 2 nautical miles (3.7km), with an average round trip time of 5ms and near zero packet loss. This meant that we were able to send video from the PTZ back to shore at > 30Hz and control the device remotely with a maximum latency of 10ms.

Shore Based Processing

A shore based control centre was used to display the main user interface on a series of large high definition screens. The operators of the system were situated in the comfortable environment of the control centre, where it was much easier to make decisions. It is also possible to fully simulate the system at the shore based installation, which is useful for development and testing.

8.3 The Pan-Tilt-Zoom Device

A significant advantage of the prototype system is that visual contact is made automatically using a PTZ camera. This means that the operators can see at a glance where objects are, what they are doing and what they look like. From a practical point of view this requires a PTZ device that can *quickly* saccade between objects and perform *precise* closed loop visual tracking. To meet these requirements we have designed, built and tested a PTZ device, which can deliver high velocities, high accelerations and accurate positioning. The design can be broken into four different areas: (i) system; (ii) mechanical; (iii) electrical and (iv) software. We will now describe the high-level system and software design, for more details regarding the mechanical and electrical design refer to Appendices E and F.

Figure 8.8 shows a system diagram of the active components that make up the PTZ device. These components can be grouped into four groups: (i) payload; (ii) pan-tilt; (iii) embedded controller and (iv) inertial sensing. We will now describe each of these groups in more detail.

Payload

The payload gives colour imagery over Gigabit Ethernet and uses a servo controlled (zoom, focus and iris) lens. The camera is a Gigabit Ethernet camera that streams uncompressed 640×480 colour video frames at up to 90fps[1]. The lens has three axes, which are DC motor driven and use potentiometers to feedback their positions. It has a horizontal field-of-view that can be varied between 35.3° at the wide end, down to 2.2° at the telephoto end, with an aspect ratio 4/3. We designed a lens controller based around an Atmel ATMega128 microprocessor that implemented Proportional-Integral-Derivative (PID) control for the three axes of the lens and communicated with the embedded PTZ controller over a proprietary serial link. The PID parameters were manually tuned to make each axis critically damped. The camera and lens assembly was calibrated at different zoom motor positions assuming a pin-hole camera model. To obtain a camera calibration matrix for a given zoom position, we fitted a parametric curve for the focal length against the motor position, which allowed us to interpolate between our calibrated zoom positions. We found that it was satisfactory to ignore radial distortion given that we were using the camera to obtain bearings of objects and to do closed loop visual control, which does not require sub-pixel level accuracy in the calibration. The payload components were housed in a waterproof enclosure for mounting on the pan-tilt device, see Figure 8.3.

Pan-Tilt

We follow the principles set out in [Murray *et al.*, 1992] and use two Harmonic Drive actuators to drive our pan-tilt. These actuators use an ingenious gearbox design to achieve: zero backlash, high gear ratios and high torques in a small package. Figure 8.4 shows an exploded assembly view of a Harmonic Drive gearbox. A gear reduction is achieved as the flexi-spline 'walks' one tooth at a time around the outer circular spline. This walking action is generated using a wave generator, which is an elliptical cam that creates a wave on its outer ring and hence in the flexi-spline. This wave has been precisely calculated to make the flexi-spline shift by one tooth at a time around the outer circular spline. The flexi-spline has a large percentage of its teeth meshed with the outer ring at any one time, which enables much larger torques to be applied than traditional gearboxes. This meshing occurs under pressure, providing zero backlash. The high torques translate into high accelerations for our pan-tilt,

[1]Typically after all visual processing has taken place we achieve frame rates of 40-60fps.

allowing us to quickly saccade the camera from one object to the next. The zero backlash meant that we could be very precise when pointing the camera. The input drive is generated using Electronically Commutated (EC) Servo motors with 8000 line incremental optical encoders to measure their position. Figure 8.5 shows the pan-tilt device and Appendix E has a selection of the mechanical drawings.

Embedded Controller

The PTZ controller consisted of: an embedded Linux processor, two motor controllers, a network hub, power distribution and power conditioning, see Figure 8.6. The embedded Linux processor ran a cut down Linux kernel with custom built hardware and drivers. It was responsible for taking in high-level commands, inertial data and motor feedback, calculating stabilisation parameters, running tight PID control loops and communicating with the motor controllers. Communication with the motor controllers was carried out using the CANOpen protocol over an industrial Controller Area Network (CAN) bus. Features of this communication protocol were: event driven prioritised messaging and heartbeat monitoring to catch any breaks in communication and to apply an emergency stop. The two motor controllers ran PID control loops for velocity and current. The motors were controlled in velocity mode as this is more suited to smooth, closed loop, visual tracking than sequential position demands. All PID loops were hand tuned starting with the motor controllers' lowest level current and velocity control loops, and finishing with the embedded Linux processors' position/velocity/stabilisation control loops.

Inertial Sensing

The system was stabilised using a 3-axis attitude and heading sensor based on MEMS inertial sensors and magnetometers. The output from the sensor was an estimate of the 3DOF orientation. Figure 8.7 shows the kinematic chain used to model the camera's motion, where: R_{wp} is the world to sensor platform transformation given by the IMU; R_{pp} is the platform to pan-tilt transformation, which describes the fixed rotation between the IMU and origin of the pan-tilt coordinate frame; R_{pc} is the pan-tilt to camera rotation given by the motor encoders and R_{wc} is the world to camera transformation, which must remain constant to stabilise the camera. The forward kinematics can be written down from Figure 8.7 as:

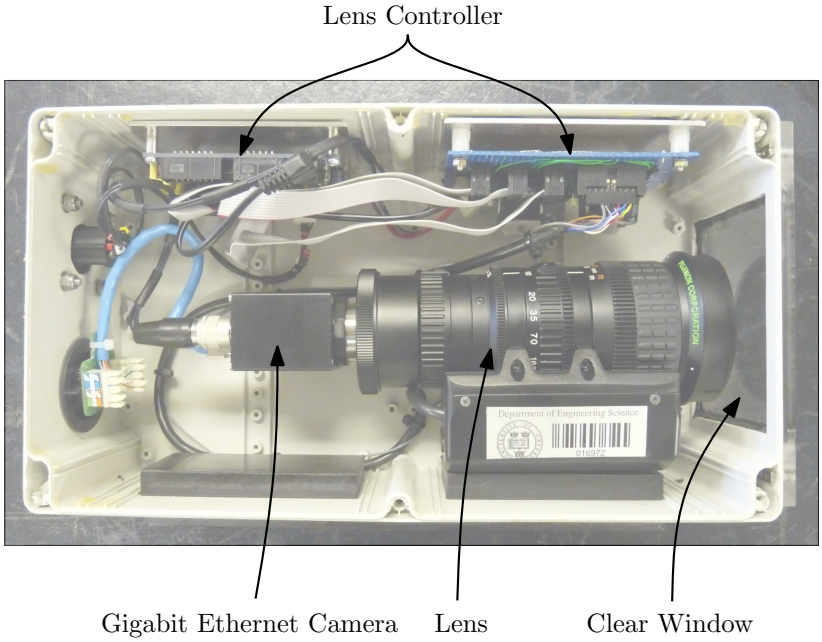


Figure 8.3: Payload with the lid off to show internal components.

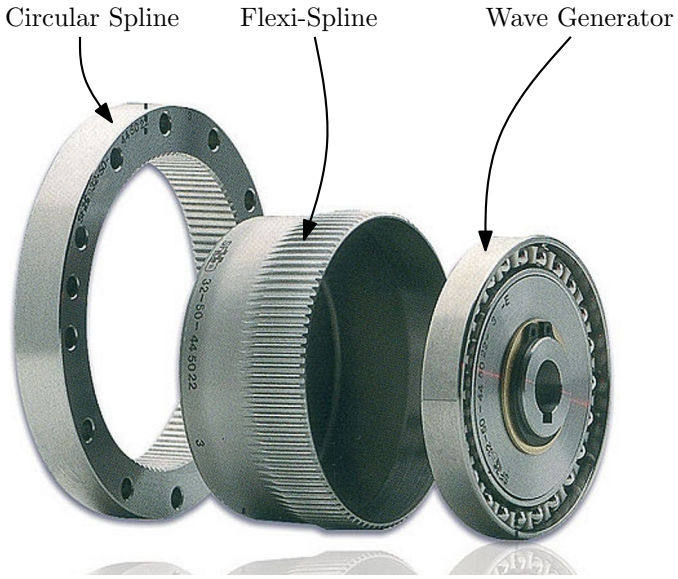


Figure 8.4: A Harmonic Drive gearbox (taken from www.directindustry.com).

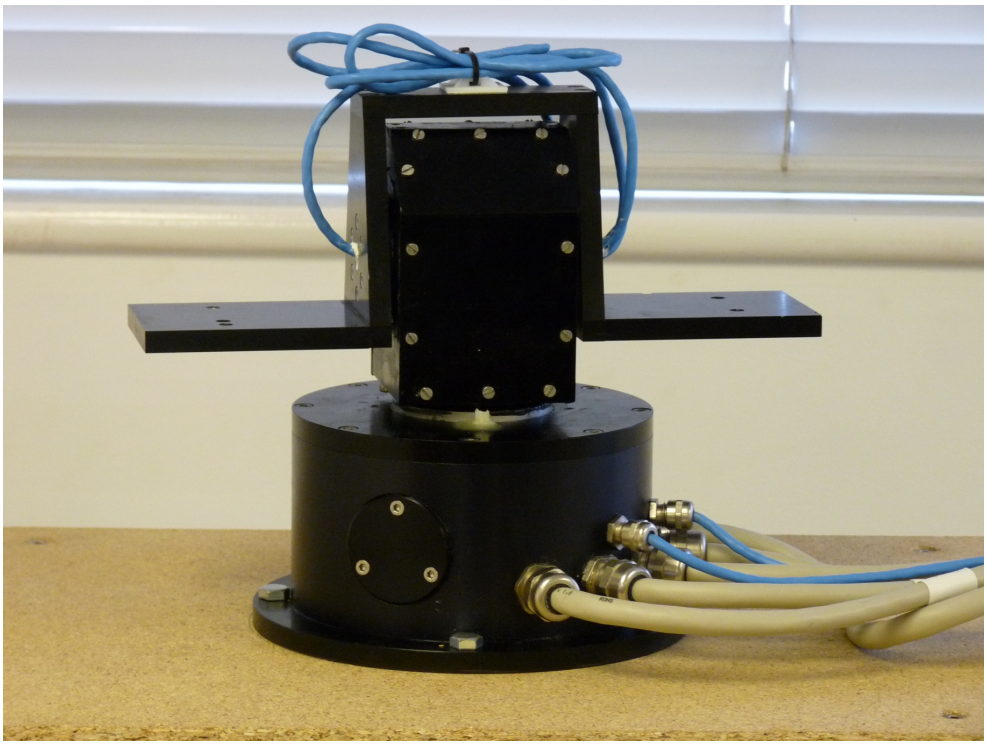


Figure 8.5: The pan-tilt device.

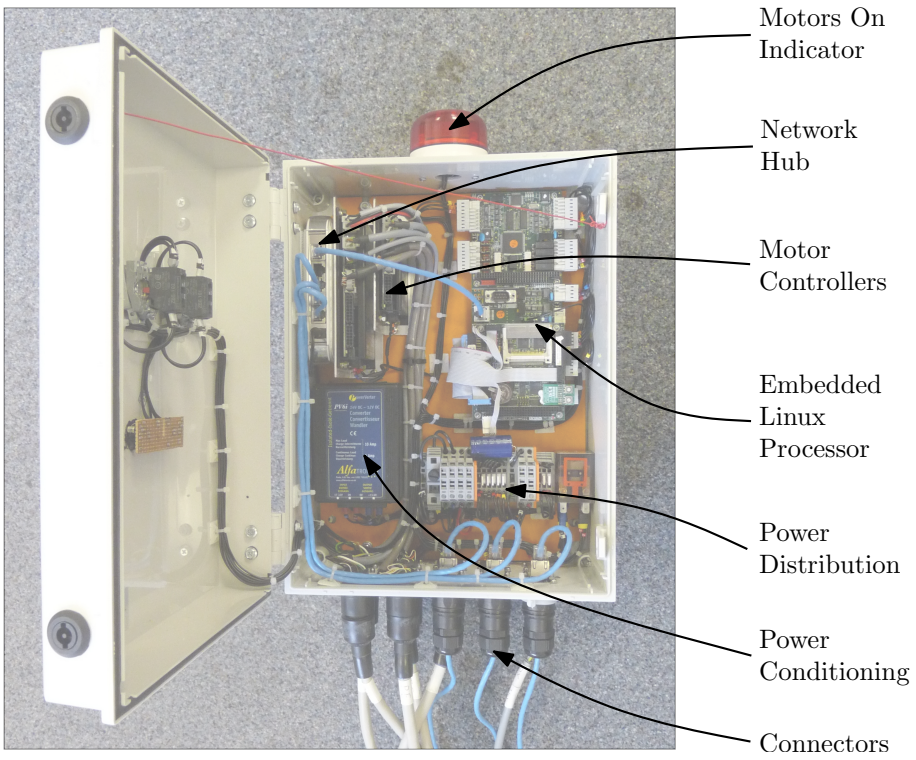


Figure 8.6: The embedded pan-tilt-zoom controller.

$$R_{wc} = R_{pc} R_{pp} R_{wp} \tag{8.1}$$

and can be rearranged to obtain the reverse kinematics:

$$R_{pc} = R_{wc} R_{wp}^T R_{pp}^T. \tag{8.2}$$

Before calculating R_{pc} any external demands, for example from the visual tracking, were first added to R_{wc}. R_{wp} was the current estimate from the IMU and R_{pp} was calibrated once in a separate experiment. After calculating R_{pc}, the pan and tilt errors were extracted from the rotation matrix and used to drive the PID control loops.

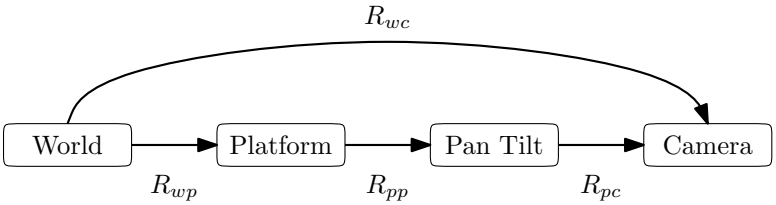


Figure 8.7: Kinematic chain.

8.4 User Interface

Figure 8.9 shows an example of the user interface. The general layout is broken into three areas: (the metric map) is displayed in the left-hand pane and shows the estimated object locations overlayed on a marine chart; (the object database) is displayed in the right-hand pane and shows a list of objects with their visual appearance and metric statistics and (the live PTZ view) is in the top-right of the left-hand pane and shows the current view from the PTZ device. The user is free to select objects either from the metric map or the object database, which means they can easily cross reference metric information to visual information and vice-versa.

The metric map shows: the vehicle's location with a cubic spline trajectory; stationary objects' locations; dynamic objects' locations with cubic spline trajectories; object IDs; landmasses; the direction of the Wi-Fi link to the shore based installation; the

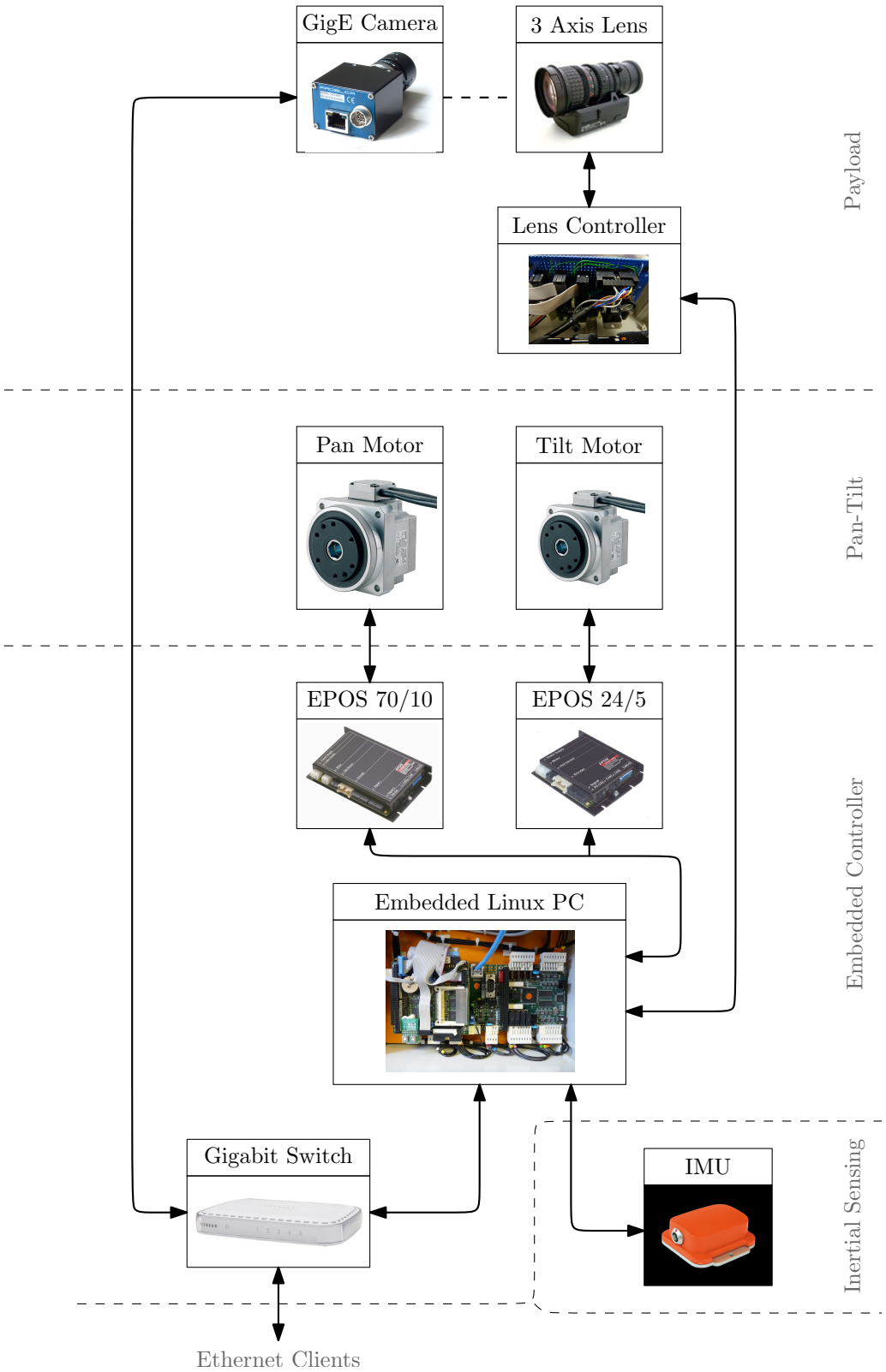


Figure 8.8: PTZ system diagram.

PTZ field of view and any user defined zones.

The object database shows each object's: ID; classification, which can be friend, foe or don't know; location, based on predefined zones; range and bearing; speed and heading; Closest Point of Approach (CPA) and Time to Closest Point of Approach (TCPA). The CPA and TCPA are useful when using the system for collision avoidance purposes as they tell you how close an object will get and when they will be closest, providing a way of shortlisting potential collision threats.

Figure 8.10 shows a close up of the live PTZ view and consists of: the pan and tilt scales, where each tick mark is one degree; a box and contour around the object being tracked; a stabilised view of the object being tracked; the current mode of operation for the visual control loop; the output from the IMU and a graphic to show the current pan and tilt values in relation to the boat's heading. During the live demonstrations, control of the PTZ could either be taken on the sensor platform or from the shore based control centre.

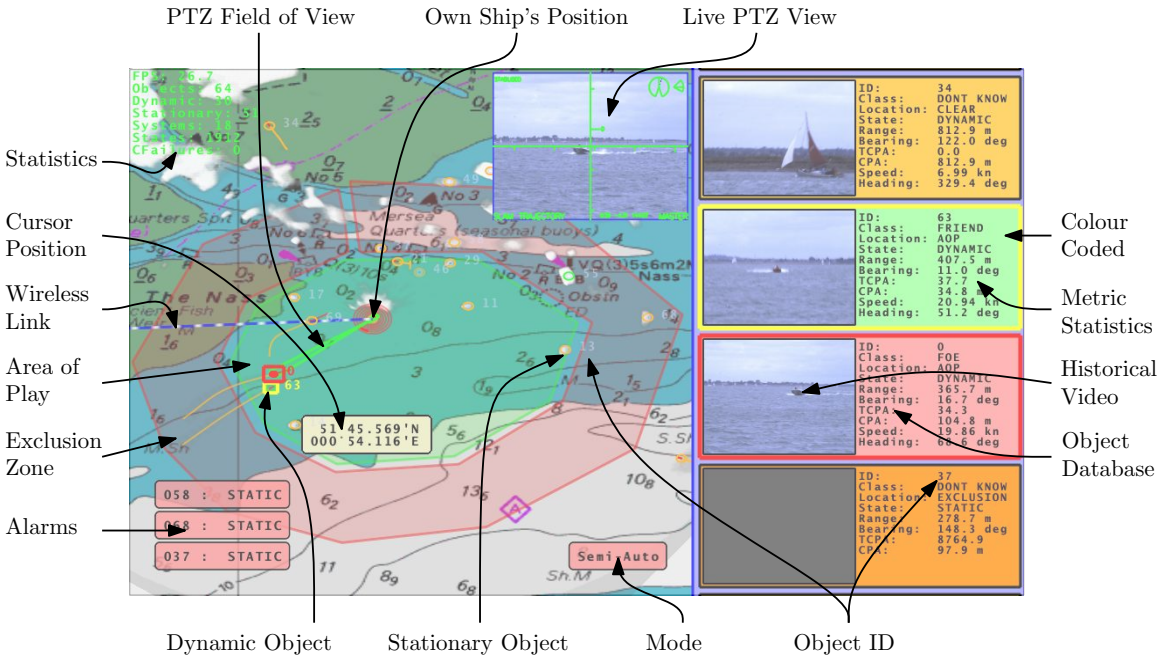


Figure 8.9: The user interface.

8.5 An Example Application

The prototype system was developed and tested over a period of three months and demonstrated live to an audience, over a period of two days. The demonstration was

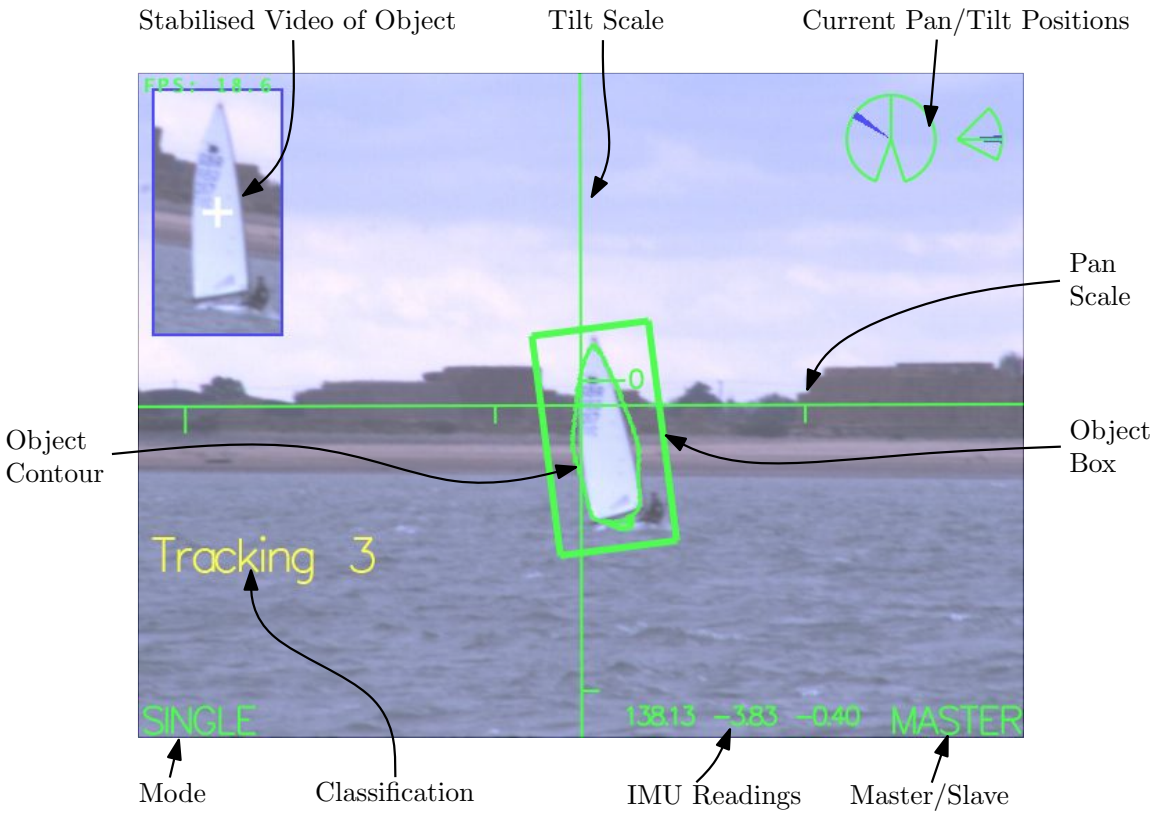


Figure 8.10: A close up of the live PTZ view.

set up to secure the entrance of a river in the Thames estuary. Figures 8.11 to 8.18 show a story board that takes the reader through an equivalent chain of events to those demonstrated. This story board has been created using the system's simulation mode to provide a clear view of how the system operates.

The aim was to protect a group of vessels operating within a pre-defined area of play, which is denoted using a green polygon, see Figure 8.11. The method for protection is to keep track of all vessels that enter a pre-defined exclusion zone, denoted with a red polygon, see Figure 8.11. Figure 8.12 shows the system a short while into the demo: the system is automatically making visual contact and populating the object database with visual information. Objects 11, 12 and 13 are all currently heading for the exclusion zone. In Figure 8.13, object 12 has reached the exclusion zone and the system generates an alarm, shown in the bottom left of the screen. The user now takes control of the system and requests a live feed of object 12; the PTZ automatically follows object 12, see Figure 8.14. The user can use this live feed to make a decision (classification) regarding the potential level of threat the object presents, see Figure 8.15. In Figure 8.16 both object 12 and 13 have been classified

as foes and a security asset (object 19) has been sent to intercept. Figure 8.17 shows the security asset intercepting the potential threats and in Figure 8.18 the user has taken manual control of the PTZ to take a closer look at the interception.

The live demonstration showed exactly this sequence of events. The audience were all situated in the control centre during the demonstration. After the rehearsed demonstration was carried out, the audience were allowed to operate the system themselves.

8.6 Conclusions

This chapter has given an insight into the practical engineering steps taken to apply the theoretical research presented in Chapters 4, 5, 6 and 7 to a real-world problem. We spent three months developing the technology from the lab into a real system, which was demonstrated live to an audience, protecting a river in the Thames Estuary from potential security threats.

The system was distributed over three geographical locations including two dynamic marine platforms i.e. boats. The whole system was networked using a combination of wired Ethernet, a high capacity 802.11n wireless link and Very High Frequency (VHF) telemetry. The data processing was carried out on everything from ATMEL microprocessors up to high performance quad core PCs. The software varied from basic PID control loops implemented in low-level C, up to probabilistic algorithms implemented in C++ and exceeded 200,000 lines of code including comments. Various mechanical and electrical components were designed, built and tested; including a high performance PTZ mechanism (some design elements are included in Appendices E and F).

In summary, as well as the theoretical contributions described in this dissertation, we have also undertaken a significant amount of practical work. This has given us the opportunity to demonstrate the use of new research technology on real-world problems. This enables us to not only scientifically validate our research, but also to demonstrate its practical applications. The next chapter will conclude the dissertation.

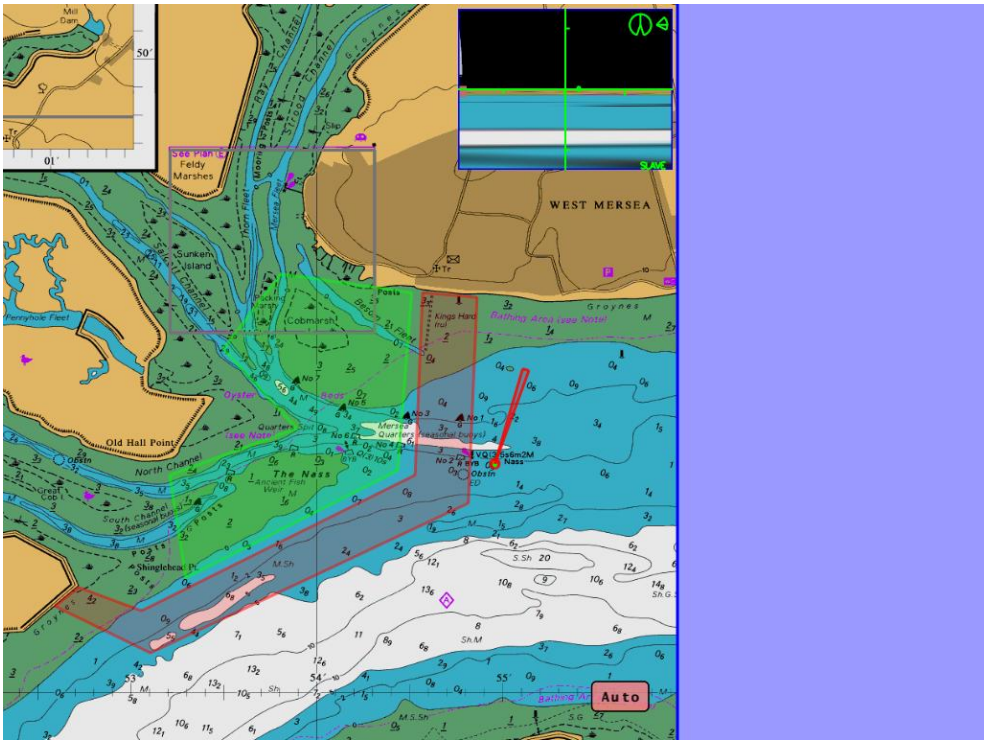


Figure 8.11: Overview of the zones: (red polygon) exclusion zone and (green polygon) area of play.

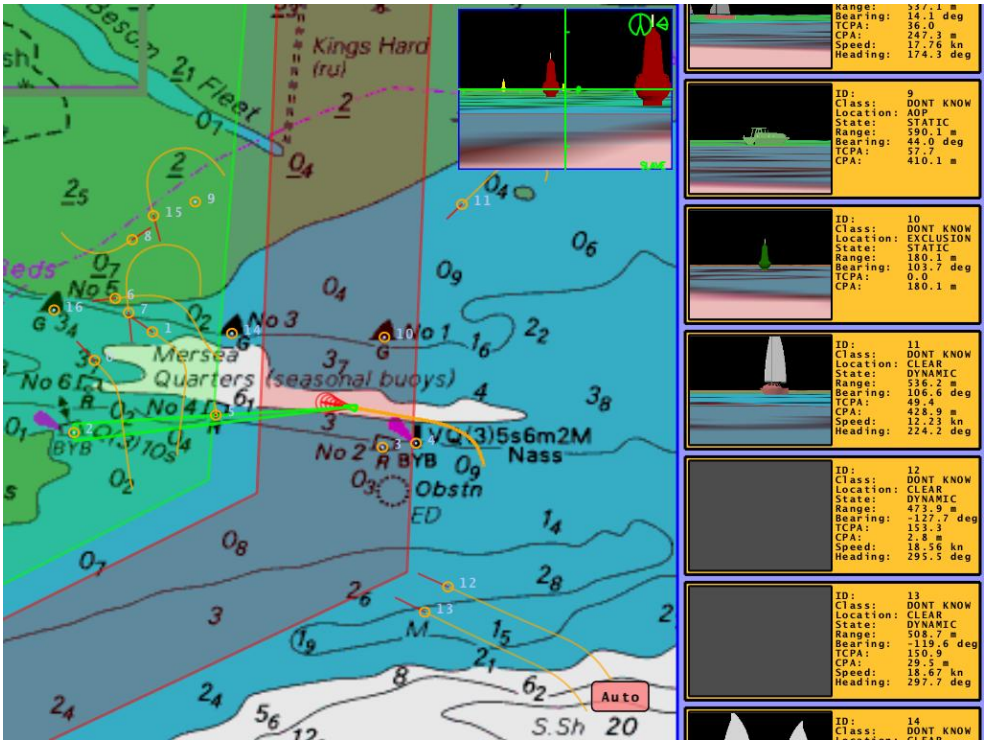


Figure 8.12: The system is automatically making visual contact and populating the object database with visual information.

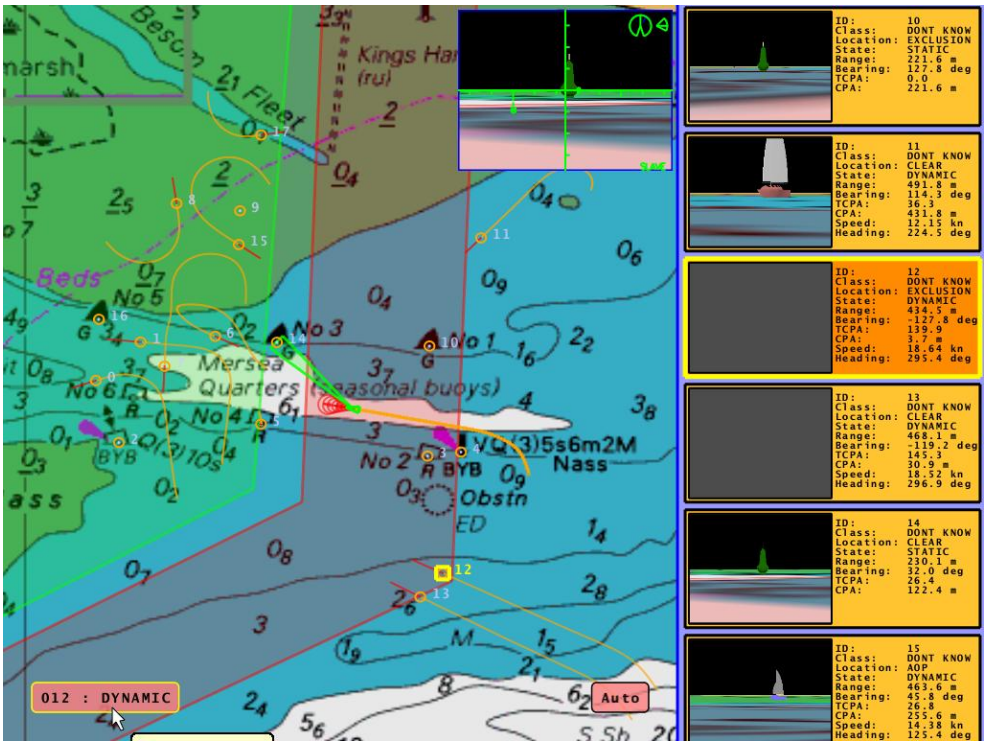


Figure 8.13: Alarm generated as an object hits the exclusion zone.

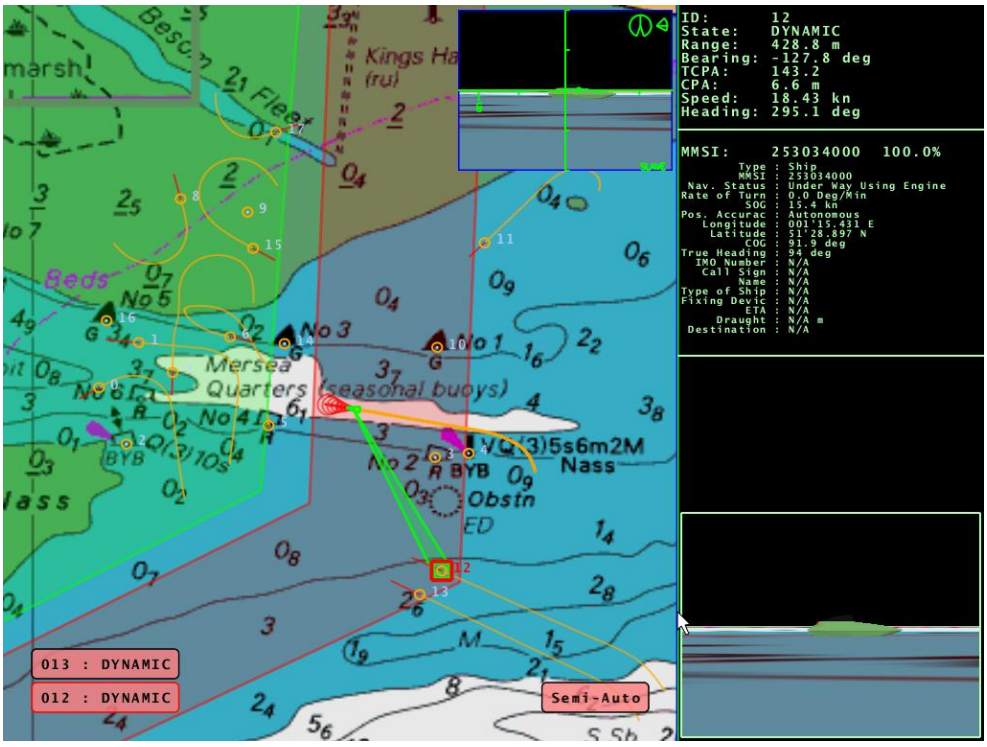


Figure 8.14: Using the system to obtain more information on the potential threat.

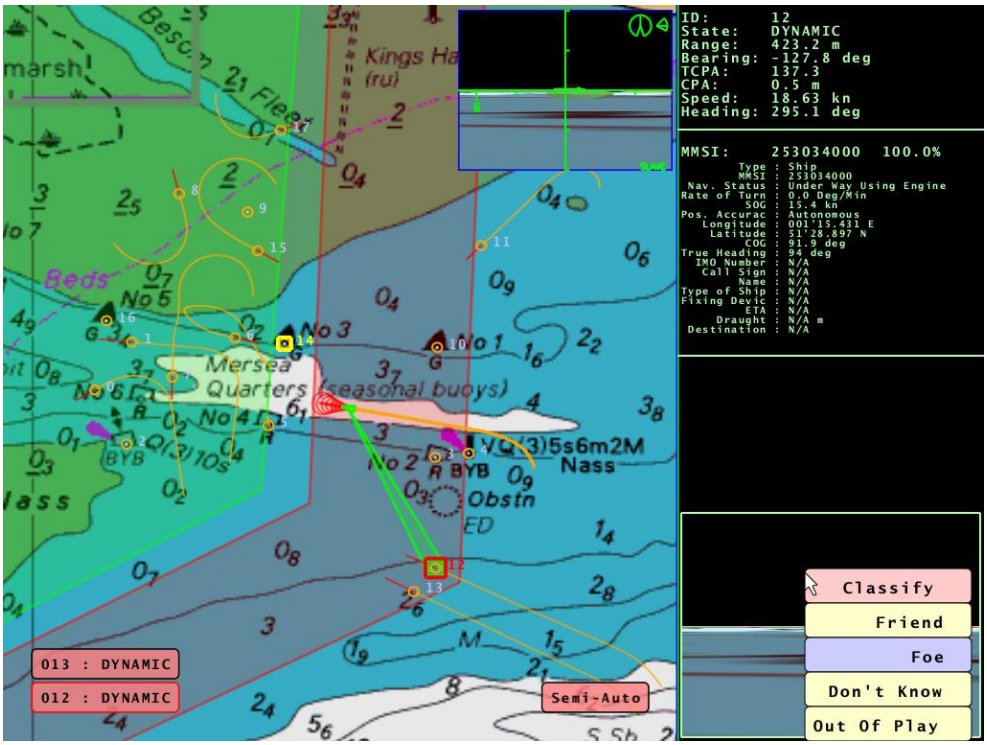


Figure 8.15: Classifying the object as a foe.

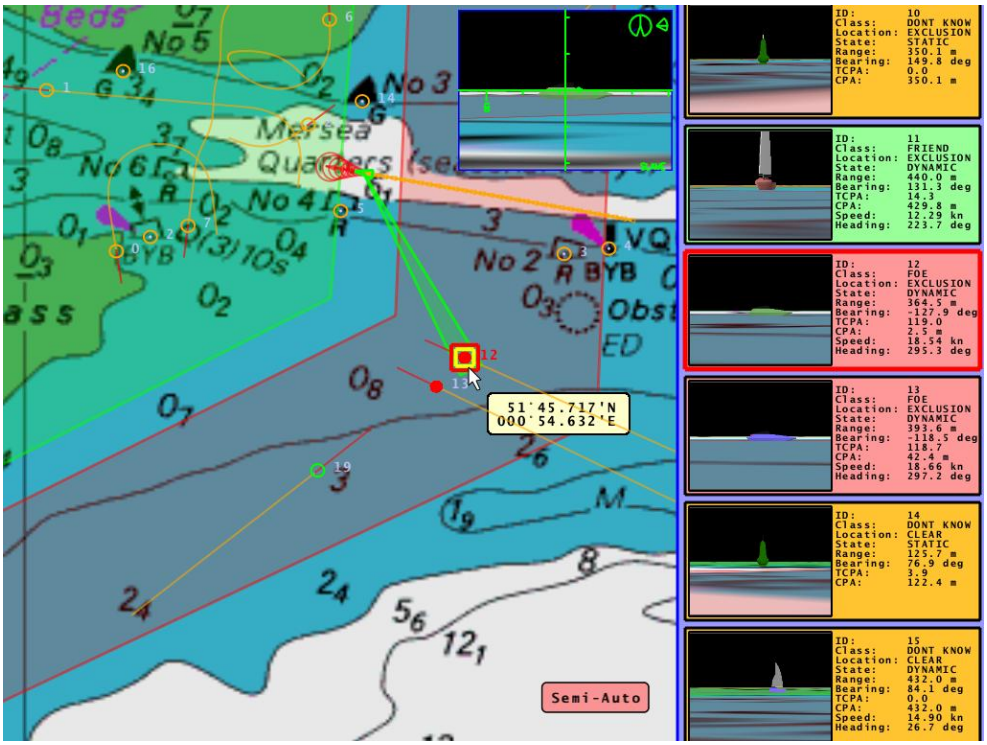


Figure 8.16: A security asset is sent to intercept.

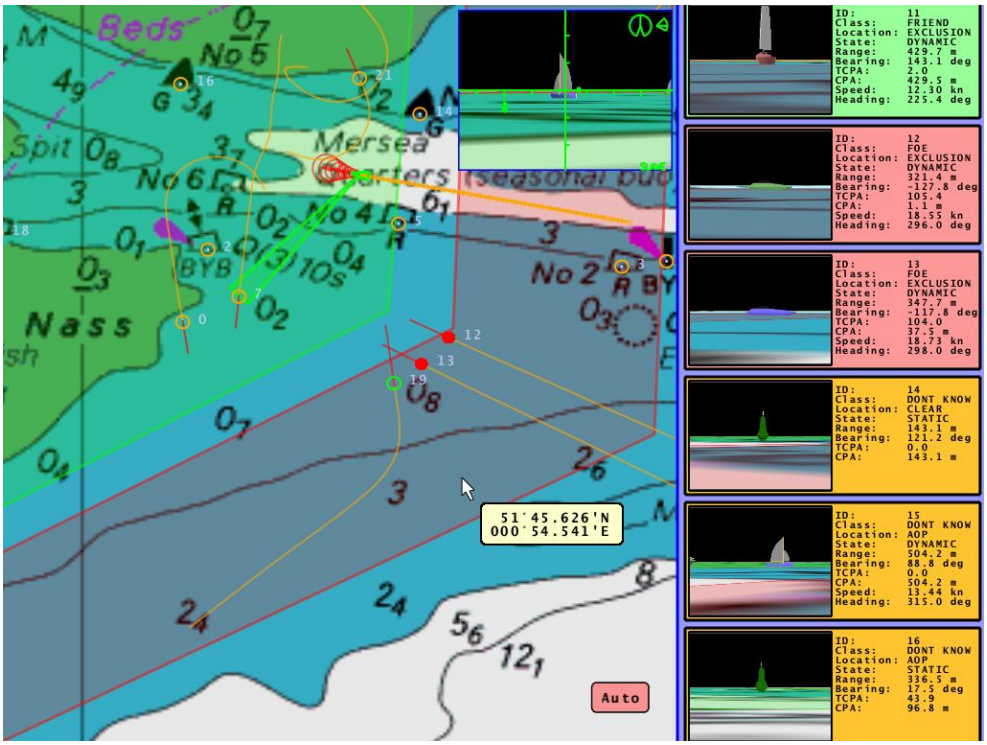


Figure 8.17: Interception.

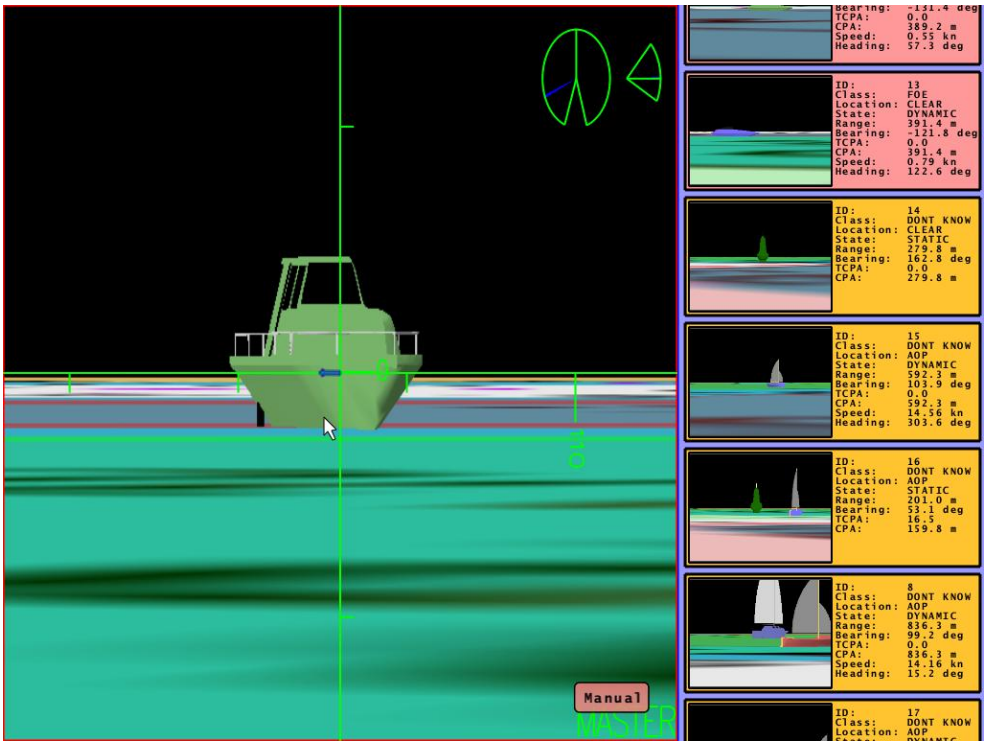


Figure 8.18: Using the PTZ to watch the interception.

Chapter 9

Conclusions

This dissertation has shown how probabilistic methods can be used to enhance situational awareness in marine environments. Specifically, we have shown how metric methods from the robotics community can be combined with visual methods from computer vision, to produce a hybrid map of the nearby environment using a mobile vehicle. This hybrid map provides users with the ability to quickly cross reference visual information (what an object looks like) with metric information (where they are and how they are moving) and vice-versa. We will now go through the specific contributions made and ideas for future work.

9.1 Estimating Metric Information

We have shown how it is possible to simultaneously estimate the position, shape and speed of stationary objects, landmasses and dynamic objects in a marine environment. The most common way to deal with dynamic objects within the robotics community is to estimate the mobile vehicle's location and then to track them separately. This has also been the common approach within the marine community. Typically GPS and compass information is used to estimate the mobile vehicle's location and then independent Kalman Filters are used to track the dynamic objects. In contrast, we have shown how dynamic objects can be incorporated directly within a SLAM framework using sliding window estimation, reversible decision making and generalised expectation maximisation. The temporal sliding window allows the system adequate time to get model-selection, data-association and clutter rejection correct before performing marginalisation. We have shown that this allows a consistent estimate to

be maintained even in the presence of dynamic objects and significant amounts of clutter.

We have also introduced a hybrid representation for the marine environment, with point features representing small stationary objects, an occupancy grid representing landmasses and cubic splines representing the trajectories of dynamic objects. This was crucial in providing a SLAM system that actually works in our real environment, where simply using point features alone was unfeasible. Part of this hybrid representation uses cubic splines to represent the trajectories of dynamic objects in the environment. This had three key benefits: (i) the number of states that need to be estimated was reduced by representing trajectories using spline sections; (ii) because the spline is continuous, it was natural to handle asynchronous measurements running at different frequencies rather than trying to interpolate and/or extrapolate measurements to match a time-step present in the estimation framework and (iii) it makes it possible to re-render sensor data at a sub-scan resolution, to account for the egomotion of the sensor platform during data acquisition.

Estimating Metric Information – Future Work

The experiments carried out for this dissertation were limited to a single sensor platform i.e. a single 10m boat. A natural course for future research would be to extend this to multiple sensor platforms, with the objective of producing a single hybrid map of the shared environment. This would involve the combination of cooperative SLAM techniques (for example [Walter & Leonard, 2004]), sliding window estimation for reversible decision making and hybrid mapping.

The current system loosely ties visual information with the estimated metric map. A more thorough approach would use this visual information to help with ambiguous data-association when estimating the metric map. Equally the metric information could be used to help disambiguate interactions/occlusions during visual tracking, by using the metric depth estimates as a prior when estimating the depth-ordering.

One potential use of the final system is in GPS denied environments. Given the hybrid map and a crude idea of where in the world you are, it is relatively straightforward to manually align the estimated data to some underlying geographical source e.g. a marine chart or satellite image. Currently this feature is supported using an intuitive user interface. However, this process could be automated using localisation techniques available in the robotics community. This would provide interesting research as the

localisation would be to a completely different type of map, created using different types of sensors, at a different time. We believe that even though the maps would be of different types, there could be enough mutual information to localise one with respect to the other.

Although we haven't discussed FAST-SLAM [Montemerlo *et al.*, 2002, 2003] yet in the dissertation, it is worth considering it as an alternative framework for doing the metric estimation. FAST-SLAM refers to the methods that use Rao-Blackwellized particle filtering to solve the SLAM problem. In particular, because of the large dimensionality of the problem, the SLAM posterior is factored into two distributions. One distribution is over the vehicle's trajectory and is represented using weighted particles and the other distribution is over the landmark locations and is represented using independent Gaussians. Each weighted particle also has its own estimate for the data-association, which allows uncertainty in data-association to be modelled. This gives significant advantages over methods that compute a single fixed solution to the data-association at each time-step. Each particle maintains an entire trajectory history (similar to Full SLAM) and yet has the computational complexity of a filtering solution. At a glance, it looks as if FAST-SLAM harnesses the benefits of a Full SLAM solution without the computational burden. However, although each particle has an entire trajectory estimate, at some point in the past all particles will share a common trajectory estimate up to that point in time (due to particle resampling). This has similarities to the temporal sliding window we use in this dissertation; however, the point at which decisions become fixed is harder to define and will be related to the number of particles and the resampling process. The ability for each particle to carry its own data-association, model-selection and map would be an alternative way of doing the estimation for SLAMIDE and would be interesting for further research.

9.2 Estimating Visual Information

We have made two contributions to visual tracking based upon our novel idea of pixel-wise posteriors, as opposed to pixel-wise likelihoods. The first method uses a simpler graphical model that can deal with a single object being tracked. The second method generalises the problem to more complicated graphical models that can handle multiple interacting/occluding objects.

We began by presenting a new visual tracking method based on pixel-wise posteriors, as opposed to pixel-wise likelihoods. Quantitative results show that this provides

better behaved objective functions for visual tracking than the current state-of-the-art. This tracking algorithm learns both the shape and appearance online and tackles the problem of tracking drift, allowing previously unseen objects to be tracked for long periods of time. We have also shown how the algorithm can be generalised using more complicated graphical models to handle multiple interacting/occluding objects. A depth posterior is computed at each time-step, allowing the tracking algorithm to account for inter-object occlusions. The method is able to simultaneously estimate the position, scale, rotation, depth-ordering, figure-ground and figure-figure segmentation for up to twelve interacting/occluding objects in *real-time*.

Estimating Visual Information – Future Work

Currently the algorithm uses implicit contours (level-sets) to represent the 2D image projection of either a 2D or 3D object. It would be interesting to see whether a 3D implicit contour representing the 3D object shape directly could be learned over time, whilst simultaneously tracking an object. Alternatively, whether a prior over the space of 2D projections for a particular object could be learned online. Either of these approaches could have the potential to improve resilience during challenging parts of a sequence, by having a tighter prior over the potential image projections an object can exhibit.

The approaches taken in this dissertation would fall into the category of a region-based tracker. An alternative and popular approach is template based tracking (see [Baker & Matthews, 2004] for an excellent summary of past work). There are situations where a template based tracker would outperform the methods proposed in this dissertation (although, they are often quite artificial). It would be very interesting to try and develop a tracking framework that combines the benefits of region-based tracking with those of template tracking. We believe that such a framework could leverage the resilience of region-based tracking and, when possible, obtain the accurate results possible with template tracking.

The current system assumes that all pixels belonging to a single object are generated from a single distribution. It would be interesting to consider learning multiple distributions for different regions of an object, for example one distribution for someone's face and another for their hair. This could be achieved using a similar framework to the multi-object tracking presented in Chapter 7. Multi-phase level-sets could be used to represent an object using more than one region and then each region could

have its own appearance model.

There are three parameters that need to be set with our method for visual tracking. They correspond to the learning rates for the: foreground model, background model and shape contour. Currently these parameters have been hand tuned once to perform best (on average) over the different types of video sequences we have. Ideally, these three parameters would change dynamically throughout a sequence to prevent learning occurring during times of confusion or if the object is lost. This would increase the resilience of the tracker.

This dissertation concentrated solely on colour imagery, a natural extension would be to use different appearance models to handle different types of image modality, for example infra-red or night vision. It would also be desirable to track on the combined image modalities e.g. colour vision plus infra-red, which could make tracking more resilient, as it would be able to capture the benefits of the different image modalities.

We currently use the OpenCV face detector [Viola & Jones, 2001] to automatically initialise the visual tracker on peoples' faces. A more useful detector for the applications of this dissertation would be a marine object detector. This would allow the system to automatically go into closed loop visual tracking.

The current system has an online appearance model that is constantly being updated to deal with any changes in appearances. The system does not keep a historic record of previous appearances. This would be a useful feature as it would enable the system to re-initialise on an object after a tracking failure or when the PTZ is sent back to look at an object it has previously observed.

The notion of an online object recognition system would be very valuable and would greatly benefit the current system. It may be possible to couple the recognition, detection and/or tracking using a common framework.

9.3 System Integration

We have shown how good practical engineering can be used to combine the theoretical contributions presented in this dissertation into a real system. The prototype system can be used for situational awareness in marine environments. It was demonstrated live over a period of two days to an audience of industry experts, governmental advisors, the navy, special forces, the police and politicians. The system was used to protect a river in the Thames Estuary against potential security threats. A com-

ponent of this system was the design, build and test of a high performance pan-tilt device, which is capable of panning 360 degrees and fixating to 1/1000th of a degree within 600ms. This type of performance is required to quickly saccade between objects in the environment so that visual information can be acquired efficiently. All of the methods and techniques that have been presented in this dissertation run in real-time i.e. 30Hz or greater on standard PC hardware.

System Integration – Future Work

One aspect of our final system was the 802.11n wireless link, which involved steerable high gain directional antennas. If the techniques presented in this dissertation were extended to multiple dynamic platforms, then there would be the need for a self organising wireless network. The 20Mbs-100Mbs link we were fortunate enough to use in this dissertation would be unavailable for more general deployments. This would create a need for producing a single hybrid map using sensors deployed on different platforms, whilst using minimal bandwidth between the platforms. This challenge alone could provide an interesting avenue for research in trying to find the optimal solution, subject to limited communications between sensor platforms. This has recently been looked at by [Nettleton *et al.*, 2003] and [Reece & Roberts, 2005].

The method for controlling the PTZ device to make automatic visual contact simply chooses the next object to look at based on the time since it was last observed. This very simple method of camera control is inefficient in terms of the amount of movement required from the PTZ. It would make an interesting research problem to first define an expression for the efficiency of the PTZ camera and secondly to solve the expression to find the optimal solution. The definition of efficiency would be dependent on different quantities, including the information in the current estimate of the environment, the amount of energy used by the PTZ, the general wear and tear of the mechanical components and which objects the user is particularly interested in. Some recent examples of work in this area are [Sommerlade & Reid, 2008] and [Soto *et al.*, 2009].

The current method for stabilising the camera uses a relatively low cost IMU sensor, which has a typical dynamic error of 2 degrees RMS. This is satisfactory only for objects that are relatively close (<1km). It would be interesting to consider how the full frame motion video could be used to help with the stabilisation problem. For instance, the horizon and/or full frame tracking could be used to help stabilisation.

This would require a very tight control loop between the visual processing and the embedded PID control loops.

The sensors and the platform were manually located in the environment to try and get the best use of sensor range and field-of-view. This could be automated either partially or fully by considering what would make the optimal sensor placement given the physical constraints.

Conclusion

We have presented a system that uses *probabilistic methods for enhanced marine situational awareness*. We have made significant contributions in the areas of SLAM and visual tracking, and demonstrated how these theoretical contributions can be applied to the practical problem of marine surveillance, see Figure 9.1. The result is not only a significant amount of academic work, but also a system that is useful for 'real-world' problems.

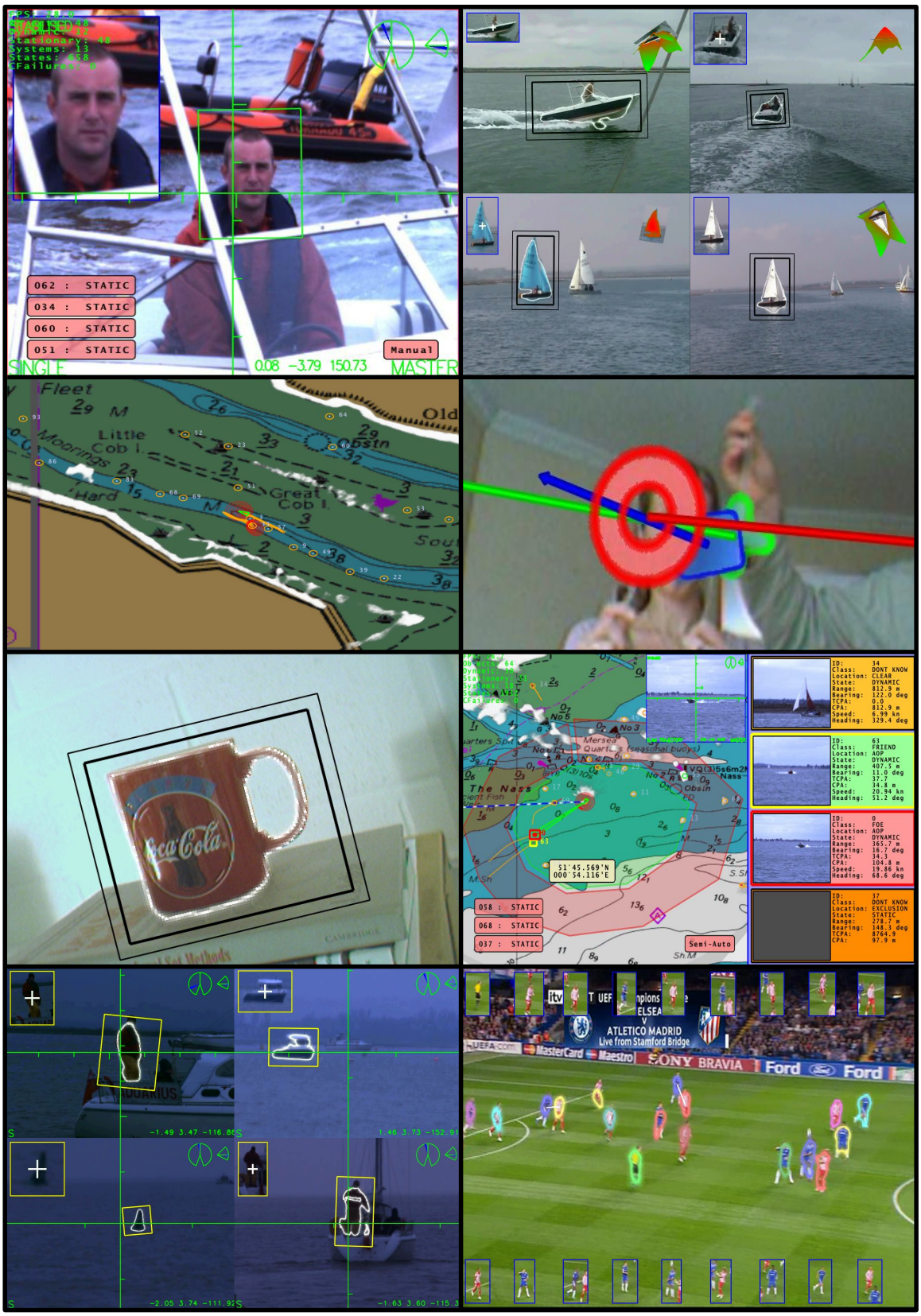


Figure 9.1: A collection of images to summarise the dissertation.

Acronyms

AIS Automatic Identification System.

CAN Controller Area Network.

CPA Closest Point of Approach.

EKF Extended Kalman Filter.

EKF-SLAM Extended Kalman Filter SLAM.

GPS Global Positioning System.

GPU Graphics Processing Unit.

HSLAMIDE Hybrid SLAM in Dynamic Environments.

IEKF Iterated Extended Kalman Filter.

IMU Inertial Measurement Unit.

JCBB Joint Compatibility Branch and Bound.

MAP Maximum a Posteriori.

MEMS Micro-Electro-Mechanical Systems.

NN Nearest Neighbour.

PC Personal Computer.

PID Proportional-Integral-Derivative.

PTZ Pan-Tilt-Zoom.

RADAR Radio Detection and Ranging.

RIB Rigid Inflatable Boat.

SLAM Simultaneous Localisation and Mapping.

SLAMIDE SLAM in Dynamic Environments.

SONAR Sound for Navigation and Ranging.

SSD Sum-of-Squared Differences.

TCPA Time to Closest Point of Approach.

UDP User Datagram Protocol.

VHF Very High Frequency.

Appendix A

Kalman Filtering Equations

Section 2.6 introduces recursive Bayesian estimation and shows how a prediction distribution (2.15) and an update distribution (2.16) can be used to probabilistically estimate the state of a system, given a sequence of measurements. We will now consider two special cases of recursive Bayesian estimation that are used extensively to model real-world processes:

A.1 The Kalman Filter

If the prediction distribution has the form:

$$P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \propto \exp(-\frac{1}{2} \|\mathbf{x}_t - \mathbf{F}_x\mathbf{x}_{t-1} - \mathbf{F}_u\mathbf{u}_t\|^2_{\mathbf{F}_u\mathbf{Q}\mathbf{F}_u^T}), \tag{A.1}$$

where $\mathbf{F}_x$ is a linear state transition matrix, describing how $\mathbf{x}_{t-1}$ will evolve to $\mathbf{x}_t$, $\mathbf{F}_u$ is the input transition matrix and $\mathbf{F}_u\mathbf{Q}\mathbf{F}_u^T$ is the input noise covariance $\mathbf{Q}$ mapped into state space; and if the measurement distribution has the form:

$$P(\mathbf{z}_t|\mathbf{x}_t) \propto \exp(-\frac{1}{2} \|\mathbf{z}_t - \mathbf{H}\mathbf{x}_t\|^2_{\mathbf{R}}), \tag{A.2}$$

where $\mathbf{H}$ is the linear measurement matrix and $\mathbf{R}$ is the covariance of the measurement noise. Then the linear Kalman Filter equations can be written down as:

Prediction:

$$\hat{\mathbf{x}}_t^* = \mathbf{F}_x \hat{\mathbf{x}}_{t-1} + \mathbf{F}_u \mathbf{u}_t \tag{A.3}$$

$$\mathbf{P}_t^* = \mathbf{F}_x \mathbf{P}_{t-1} \mathbf{F}_x^T + \mathbf{F}_u \mathbf{Q} \mathbf{F}_u^T \tag{A.4}$$

Update:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^* + \mathbf{W}\mathbf{v} \tag{A.5}$$

$$\mathbf{P}_t = \mathbf{P}_t^* - \mathbf{W}\mathbf{S}\mathbf{W}^T \tag{A.6}$$

where:

$$\mathbf{v} = \mathbf{z}_t - \mathbf{H}\hat{\mathbf{x}}_t^* \tag{A.7}$$

$$\mathbf{S} = \mathbf{H}\mathbf{P}_t^* \mathbf{H}^T + \mathbf{R} \tag{A.8}$$

$$\mathbf{W} = \mathbf{P}_t^* \mathbf{H}^T \mathbf{S}^{-1} \tag{A.9}$$

where $\hat{\mathbf{x}}_{t-1}$ represents the previous estimate, $\hat{\mathbf{x}}_t^*$ represents the predicted estimate and $\hat{\mathbf{x}}_t$ represents the updated estimate. Likewise, $\mathbf{P}_{t-1}$ represents the covariance of the previous estimate, $\mathbf{P}_t^*$ represents the covariance of the predicted estimate and $\mathbf{P}_t$ represents the covariance of the updated estimate.

A.2 The Extended Kalman Filter

If the prediction distribution has the form:

$$P(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_t) \propto \exp(-\frac{1}{2} \left\| \mathbf{x}_t - f(\mathbf{x}_{t-1}, \mathbf{u}_t) \right\|^2_{\mathbf{\Delta F}_u \mathbf{Q} \mathbf{\Delta F}_u^T}), \tag{A.10}$$

where $f(\ldots)$ is the non-linear state transition model and the measurement distribution has the form:

$$P(\mathbf{z}_t|\mathbf{x}_t) \propto \exp(-\frac{1}{2} \left\| \mathbf{z}_t - h(\mathbf{x}_t) \right\|^2_{\mathbf{R}}), \tag{A.11}$$

where $h(\ldots)$ is the non-linear measurement model. Then the equations corresponding to the EKF are:

Prediction:

$$\hat{\mathbf{x}}_t^* = f(\hat{\mathbf{x}}_{t-1}, \mathbf{u}_t) \tag{A.12}$$

$$\mathbf{P}_t^* = \mathbf{\Delta F}_x \mathbf{P}_{t-1} \mathbf{\Delta F}_x^T + \mathbf{\Delta F}_u \mathbf{Q} \mathbf{\Delta F}_u^T \tag{A.13}$$

Update:

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_t^* + \mathbf{W}\mathbf{v} \tag{A.14}$$

$$\mathbf{P}_t = \mathbf{P}_t^* - \mathbf{W}\mathbf{S}\mathbf{W}^T \tag{A.15}$$

where:

$$\mathbf{v} = \mathbf{z}_t - h(\hat{\mathbf{x}}_t^*) \tag{A.16}$$

$$\mathbf{S} = \mathbf{\Delta H} \mathbf{P}_t^* \mathbf{\Delta H}^T + \mathbf{R} \tag{A.17}$$

$$\mathbf{W} = \mathbf{P}_t^* \mathbf{\Delta H}^T \mathbf{S}^{-1} \tag{A.18}$$

where the $\mathbf{\Delta}$ symbol represents the Jacobian e.g. $\mathbf{\Delta F}_x$ is the Jacobian of $f(\ldots)$ with respect to $\mathbf{x}$ and $\mathbf{\Delta F}_u$ is the Jacobian of $f(\ldots)$ with respect to $\mathbf{u}$.

Appendix B

Least-Squares SLAM Example

Section 3.7 describes least-squares SLAM. We will now give an example of the structure of Equation (3.19). Let us consider the structure of $\mathbf{A}$, $\boldsymbol{\Sigma}^{-1}$, $\mathbf{b}$ and $\boldsymbol{\delta}$ for a simple example with four time-steps $\{t = 0, 1, 2, 3\}$, where the vehicle observes landmarks $\{\mathbf{m}_0, \mathbf{m}_1, \mathbf{m}_0\}$ at time-steps $\{t = 1, 2, 3\}$ respectively:

$$\mathbf{A} = \left[\begin{array}{cccccc} & & & -\mathbf{I} & & \\ \hline \mathbf{F}_1 & -\mathbf{I} & & & & \\ & \mathbf{F}_2 & -\mathbf{I} & & & \\ & & \mathbf{F}_3 & -\mathbf{I} & & \\ & \mathbf{H}_1 & & & \mathbf{J}_1 & \\ & & \mathbf{H}_2 & & & \mathbf{J}_2 \\ & & & \mathbf{H}_3 & \mathbf{J}_3 & \end{array}\right] \tag{B.1}$$

$$\boldsymbol{\Sigma}^{-1} = \left[\begin{array}{ccccccc} \boldsymbol{\Pi}^{-1} & & & & & & \\ & \mathbf{Q}_1^{-1} & & & & & \\ & & \mathbf{Q}_2^{-1} & & & & \\ & & & \mathbf{Q}_3^{-1} & & & \\ & & & & \mathbf{R}_1^{-1} & & \\ & & & & & \mathbf{R}_2^{-1} & \\ & & & & & & \mathbf{R}_3^{-1} \end{array}\right] \tag{B.2}$$

$$\mathbf{b} = \begin{bmatrix} \mathbf{x} - \tilde{\mathbf{x}} \\ \mathbf{x}_1 - f(\mathbf{x}_0, \mathbf{u}_1) \\ \mathbf{x}_2 - f(\mathbf{x}_1, \mathbf{u}_2) \\ \mathbf{x}_3 - f(\mathbf{x}_2, \mathbf{u}_3) \\ \mathbf{z}_1 - h(\mathbf{x}_1, \mathbf{M}) \\ \mathbf{z}_2 - h(\mathbf{x}_2, \mathbf{M}) \\ \mathbf{z}_3 - h(\mathbf{x}_3, \mathbf{M}) \end{bmatrix} \qquad \boldsymbol{\delta} = \begin{bmatrix} \boldsymbol{\delta}\mathbf{x}_0 \\ \boldsymbol{\delta}\mathbf{x}_1 \\ \boldsymbol{\delta}\mathbf{x}_2 \\ \boldsymbol{\delta}\mathbf{x}_3 \\ \boldsymbol{\delta}\mathbf{m}_0 \\ \boldsymbol{\delta}\mathbf{m}_1 \end{bmatrix} \tag{B.3}$$

Appendix C

Calculus of Variations

We will now show how to differentiate the summation term in Equation (6.11) in Chapter 6:

$$\sum_{i=1}^{N}\left\{\log\left(P(\mathbf{x}_i|\mathbf{\Phi},\mathbf{p},\mathbf{y}_i)\right)-\frac{(|\nabla\mathbf{\Phi}(\mathbf{x}_i)|-1)^2}{2\sigma^2}\right\} \tag{C.1}$$

where

$$P(\mathbf{x}_i|\mathbf{\Phi},\mathbf{p},\mathbf{y}_i)=\frac{P_f H_\epsilon(\mathbf{\Phi})+P_b(1-H_\epsilon(\mathbf{\Phi}))}{P_f\sum H_\epsilon(\mathbf{\Phi})+P_b\sum(1-H_\epsilon(\mathbf{\Phi}))}$$

and

$$P_f = P(\mathbf{y}_i|M_f),\qquad P_b=P(\mathbf{y}_i|M_b),\qquad \mathbf{\Phi}=\mathbf{\Phi}(\mathbf{x}_i),$$

with respect to $\mathbf{\Phi}$ using calculus of variations [Evans, 2002] to express the first variation (Gateaux derivative) of the functional as:

$$\frac{\partial\log(P(\mathbf{\Phi},\mathbf{p}|\mathbf{\Omega}))}{\partial\mathbf{\Phi}}=\frac{\delta_\epsilon(\mathbf{\Phi})(P_f-P_b)}{P_f H_\epsilon(\mathbf{\Phi})+P_b(1-H_\epsilon(\mathbf{\Phi}))}-\frac{1}{\sigma^2}\left[\nabla^2\mathbf{\Phi}-\mathrm{div}\left(\frac{\nabla\mathbf{\Phi}}{|\nabla\mathbf{\Phi}|}\right)\right], \tag{C.2}$$

where ∇^2 is the Laplacian operator and $\delta_\epsilon(\mathbf{\Phi})$ is the derivative of the blurred Heaviside step function, i.e. a blurred Dirac delta function. Let us begin with the following differential:

$$\frac{\partial \log\left(P(\mathbf{x}_i|\mathbf{\Phi},\mathbf{p},\mathbf{y}_i)\right)}{\partial \mathbf{\Phi}} = \frac{1}{P(\mathbf{x}_i|\mathbf{\Phi},\mathbf{p},\mathbf{y}_i)} \frac{\partial P(\mathbf{x}_i|\mathbf{\Phi},\mathbf{p},\mathbf{y}_i)}{\partial \mathbf{\Phi}} \tag{C.3}$$

where

$$\frac{\partial P(\mathbf{x}_i|\mathbf{\Phi},\mathbf{p},\mathbf{y}_i)}{\partial \mathbf{\Phi}} = \frac{\left\{P_f \sum H_\epsilon(\mathbf{\Phi}) + P_b \sum (1 - H_\epsilon(\mathbf{\Phi})) - P_f H_\epsilon(\mathbf{\Phi}) - P_b(1 - H_\epsilon(\mathbf{\Phi}))\right\} \delta_\epsilon(\mathbf{\Phi})(P_f - P_b)}{\left\{P_f \sum H_\epsilon(\mathbf{\Phi}) + P_b \sum (1 - H_\epsilon(\mathbf{\Phi}))\right\}^2}. \tag{C.4}$$

Substituting (C.4) back in to (C.3) and simplifying gives:

$$\frac{\partial \log\left(P(\mathbf{x}_i|\mathbf{\Phi},\mathbf{p},\mathbf{y}_i)\right)}{\partial \mathbf{\Phi}} = \frac{\delta_\epsilon(\mathbf{\Phi})(P_f - P_b)}{P_f H_\epsilon(\mathbf{\Phi}) + P_b(1 - H_\epsilon(\mathbf{\Phi}))} - \frac{\delta_\epsilon(\mathbf{\Phi})(P_f - P_b)}{P_f \sum H_\epsilon(\mathbf{\Phi}) + P_b \sum (1 - H_\epsilon(\mathbf{\Phi}))}. \tag{C.5}$$

Using the following three conditions:

$$\begin{aligned} H_\epsilon(\mathbf{\Phi}) &>= 0 \\ \sum H_\epsilon(\mathbf{\Phi}) &>> H_\epsilon(\mathbf{\Phi}) \\ \sum (1 - H_\epsilon(\mathbf{\Phi})) &>> (1 - H_\epsilon(\mathbf{\Phi})), \end{aligned} \tag{C.6}$$

we can approximate (C.5) by dropping the second term:

$$\frac{\partial \log\left(P(\mathbf{x}_i|\mathbf{\Phi},\mathbf{p},\mathbf{y}_i)\right)}{\partial \mathbf{\Phi}} = \frac{\delta_\epsilon(\mathbf{\Phi})(P_f - P_b)}{P_f H_\epsilon(\mathbf{\Phi}) + P_b(1 - H_\epsilon(\mathbf{\Phi}))}. \tag{C.7}$$

We have now differentiated the first term in (C.1). The second term:

$$\frac{(|\nabla \mathbf{\Phi}| - 1)^2}{2\sigma^2} \tag{C.8}$$

can be differentiated with respect to $\mathbf{\Phi}$ using the Euler-Lagrange equation:

$$\frac{\partial f}{\partial \mathbf{\Phi}} = \nabla \cdot \frac{\partial f}{\partial \nabla \mathbf{\Phi}} \tag{C.9}$$

and the following two identities:

$$|\nabla \mathbf{\Phi}| = \sqrt{\nabla \mathbf{\Phi} \cdot \nabla \mathbf{\Phi}} \tag{C.10}$$

and

$$\frac{\partial |\nabla \mathbf{\Phi}|}{\partial \nabla \mathbf{\Phi}} = \frac{\nabla \mathbf{\Phi}}{|\nabla \mathbf{\Phi}|}. \tag{C.11}$$

Substituting (C.8) into (C.9) gives us:

$$\begin{aligned} \frac{\partial \frac{(|\nabla \mathbf{\Phi}|-1)^2}{2\sigma^2}}{\partial \mathbf{\Phi}} &= \nabla \cdot \frac{(|\nabla \mathbf{\Phi}| - 1)}{\sigma^2} \frac{\nabla \mathbf{\Phi}}{|\nabla \mathbf{\Phi}|} \\ &= \frac{1}{\sigma^2} \left[\nabla^2 \mathbf{\Phi} - \operatorname{div} \left(\frac{\nabla \mathbf{\Phi}}{|\nabla \mathbf{\Phi}|} \right) \right]. \end{aligned} \tag{C.12}$$

Appendix D

Prototype System – Design Concept

Figure D.1 shows a mock up of what the shore based control centre for a port might look like. At the front of the room is a tactical display that summarises the general situation and provides those in charge with enough information to assign particular problems to individual operators. The operators each have a workstation, which allows them to concentrate on a smaller subset of the information being processed by the system, so that they can make informed decisions about how to deal with a particular problem. To complement the shore based facility there would also be several security assets afloat, for example small RIBs or harbour patrol vessels. Each one of these security assets would have a similar system aboard, that gives them access to the situational awareness information most relevant to their own tasks i.e. details of nearby objects.

Figure D.2 shows a close up of the tactical display in the shore based station. The centre of the screen shows the system's current estimate of what is going on in the port: the green symbols represent known objects that are behaving correctly; the red symbols represent unknown objects that may pose a security threat to the port and the yellow symbols represent the security assets that are under the control of the port security team. The sides of the screen show live videos of interesting objects, which are automatically acquired using distributed cameras and smart visual tracking techniques. In the example in Figure D.2, there are two unknown objects that could pose a threat to the port, each of these would be assigned to one of the operators to deal with.

Figure D.1: The control room with the tactical display at the front and four operators' workstations.

Figure D.3 shows the view of an operator's workstation. In this particular case, the operator has been assigned a security threat to deal with and they have used the system's smart visual tracking capabilities to acquire enough visual information to identify the object as suspicious. Now that the operator has classified the object, he can assign a security asset (harbour patrol vessel) to go and question the security threat's intent. The assignment and instruction of how to get to the security threat are automatically transferred to the security asset, so that the planning can be done in the comfort of the shore based installation rather than on a boat, which may already be dealing with difficult conditions.

Figure D.4 shows the view from within a security asset, where the information regarding the security threat has automatically been transferred to their system. They can see what the security threat looks like and route information on how to navigate to it is automatically transferred to their navigation system, so they do not have to worry about planning the journey.

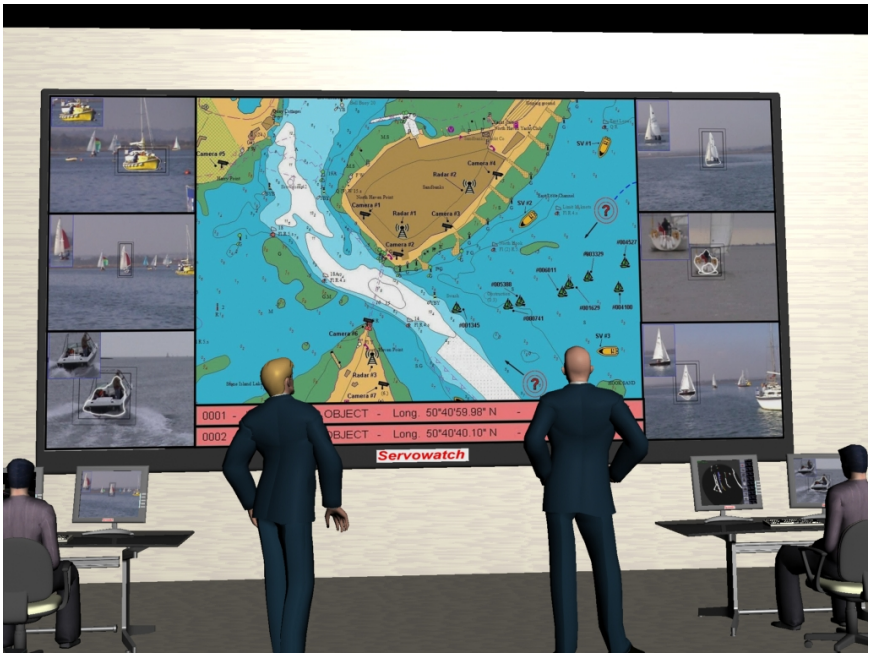


Figure D.2: The tactical display.

Figure D.3: An operator's workstation.

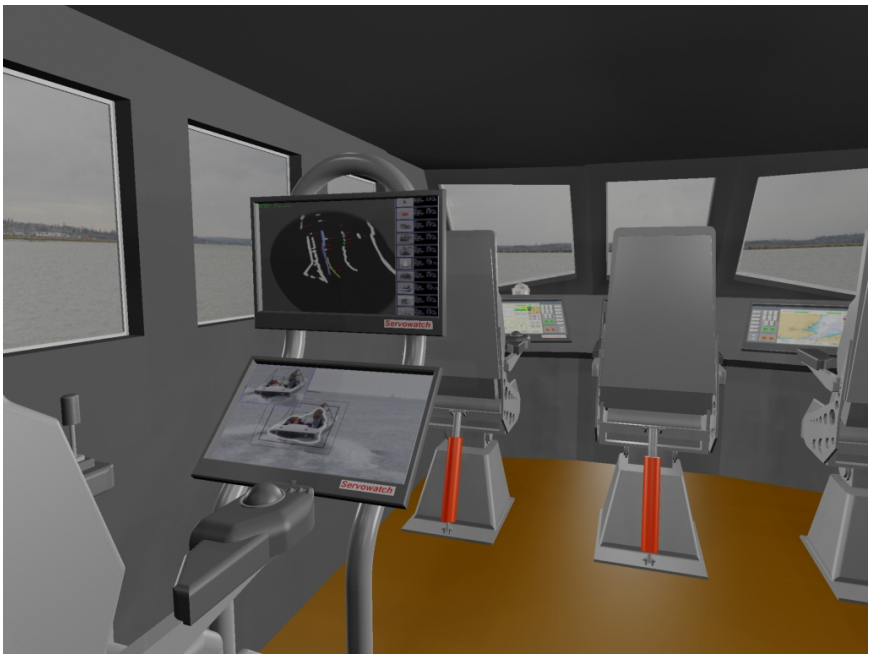
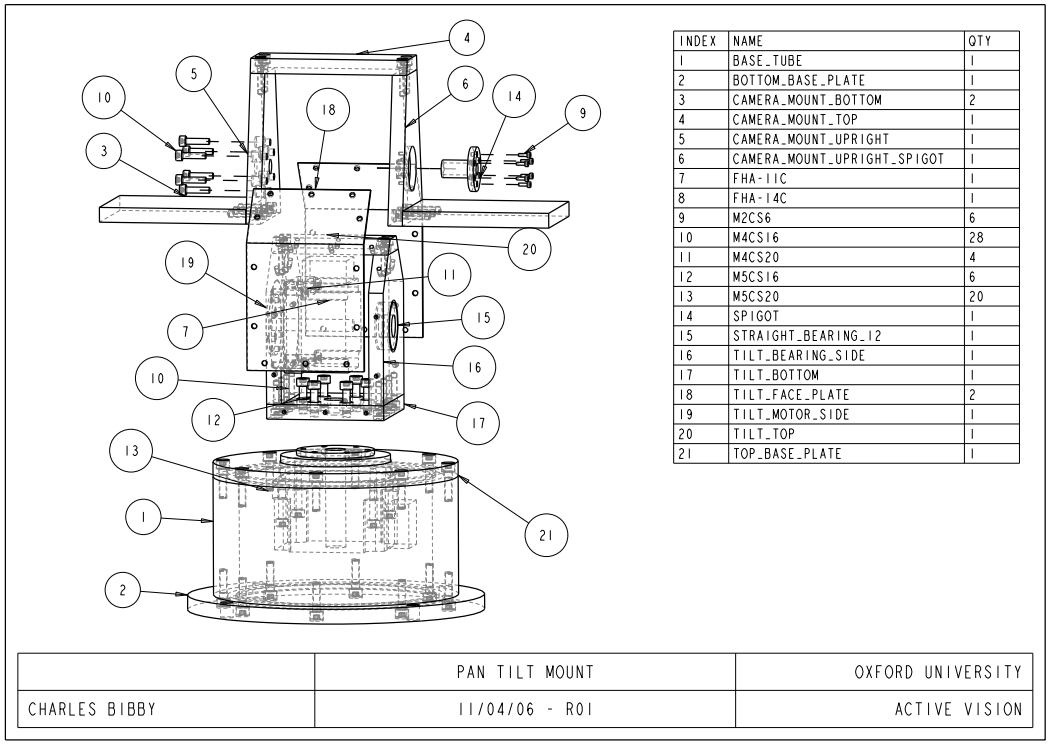


Figure D.4: A view from inside one of the security assets.

Appendix E

PTZ Design – Mechanical



INDEX	NAME	QTY
1	BASE_TUBE	1
2	BOTTOM_BASE_PLATE	1
3	CAMERA_MOUNT_BOTTOM	2
4	CAMERA_MOUNT_TOP	1
5	CAMERA_MOUNT_UPRIGHT	1
6	CAMERA_MOUNT_UPRIGHT_SPIGOT	1
7	FHA-11C	1
8	FHA-14C	1
9	M2CS6	6
10	M4CSI6	28
11	M4CS20	4
12	M5CSI6	6
13	M5CS20	20
14	SPIGOT	1
15	STRAIGHT_BEARING_12	1
16	TILT_BEARING_SIDE	1
17	TILT_BOTTOM	1
18	TILT_FACE_PLATE	2
19	TILT_MOTOR_SIDE	1
20	TILT_TOP	1
21	TOP_BASE_PLATE	1

	PAN TILT MOUNT	OXFORD UNIVERSITY
CHARLES BIBBY	11/04/06 - R01	ACTIVE VISION

Figure E.1: Assembly – Pan-Tilt.

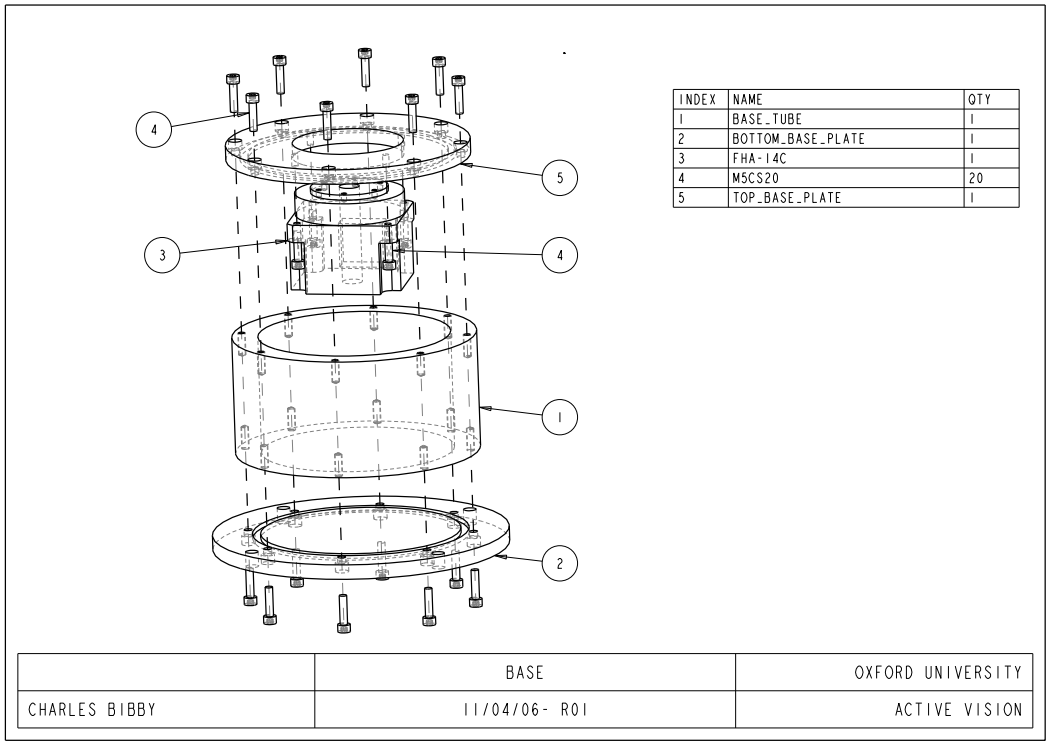


Figure E.2: Assembly – Base.

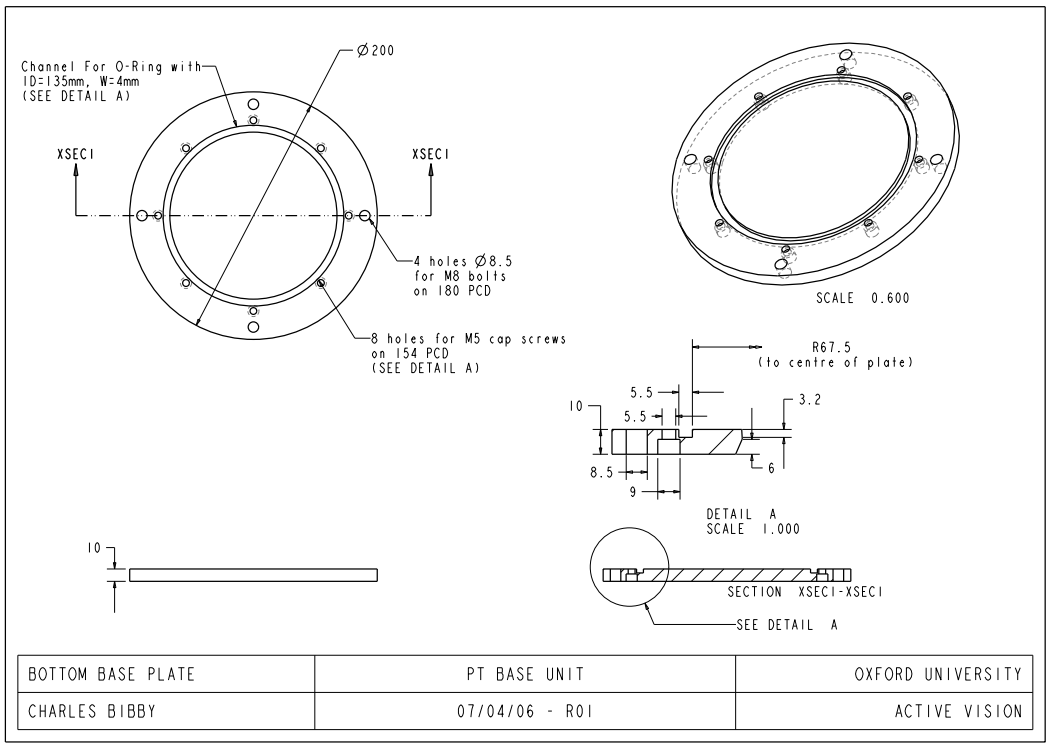
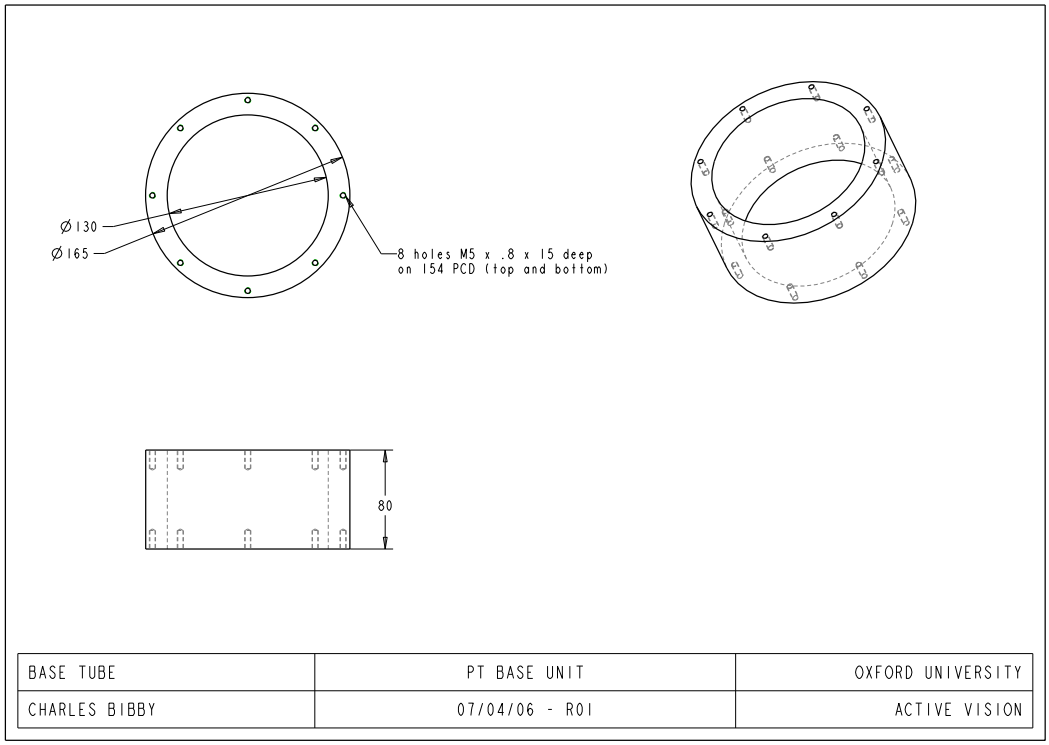
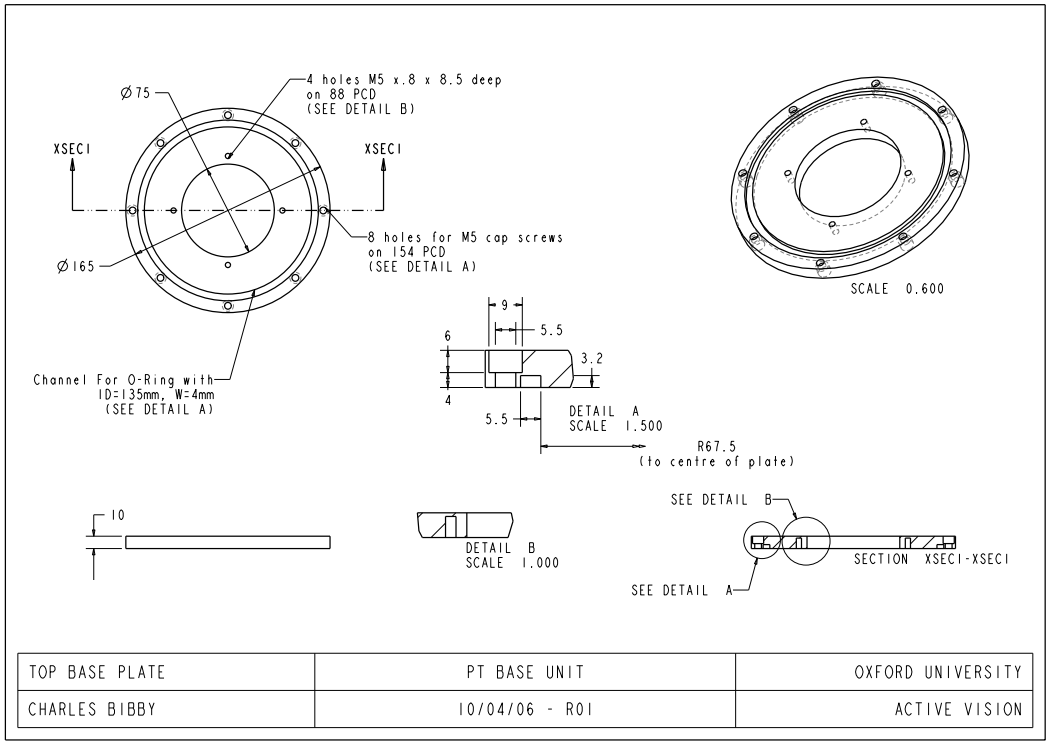


Figure E.3: Part – Base – Bottom Plate.



BASE TUBE	PT BASE UNIT	OXFORD UNIVERSITY
CHARLES BIBBY	07/04/06 - R01	ACTIVE VISION

Figure E.4: Part – Base – Tube.



TOP BASE PLATE	PT BASE UNIT	OXFORD UNIVERSITY
CHARLES BIBBY	10/04/06 - R01	ACTIVE VISION

Figure E.5: Part – Base – Top Plate.

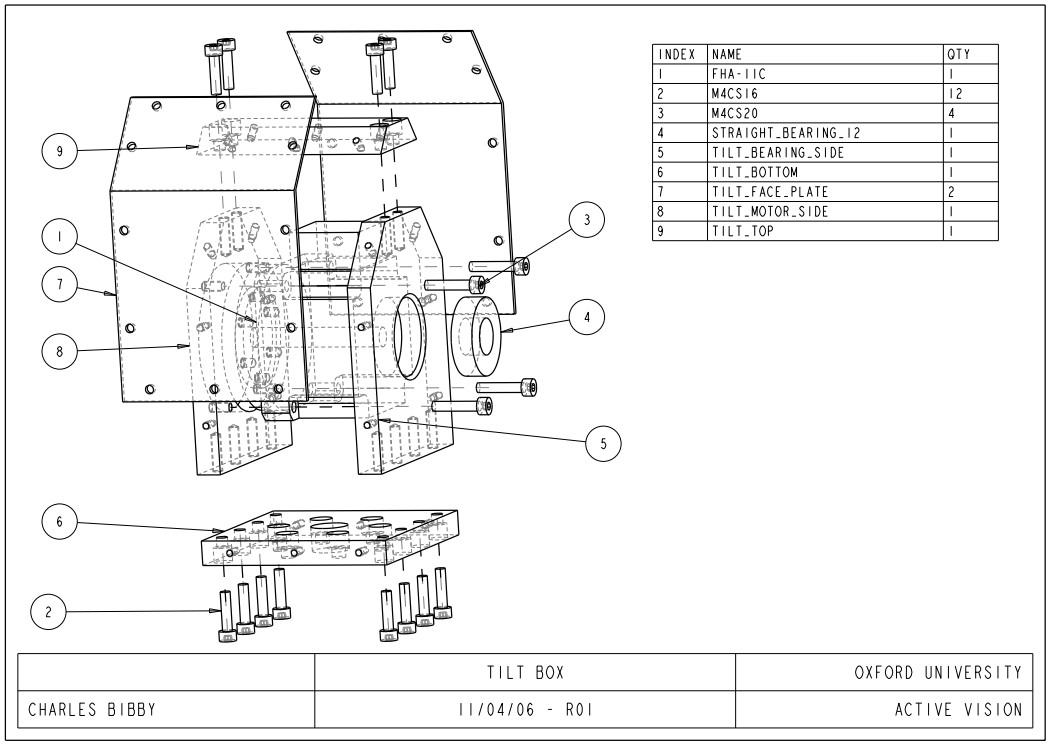


Figure E.6: Assembly – Tilt Box.

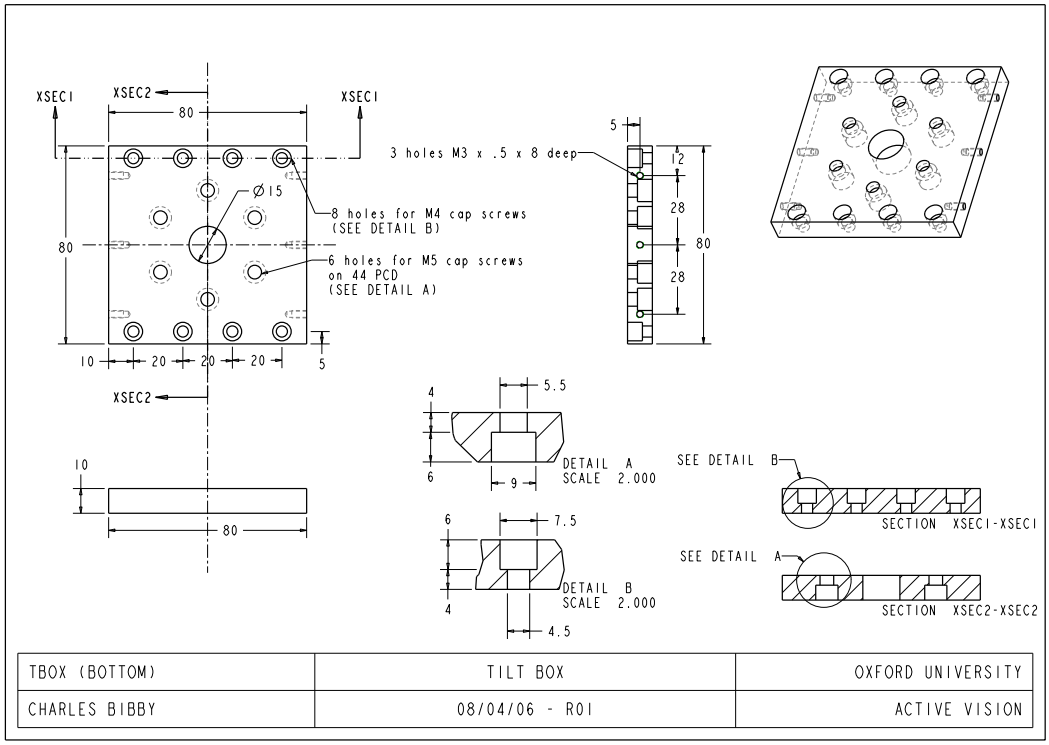
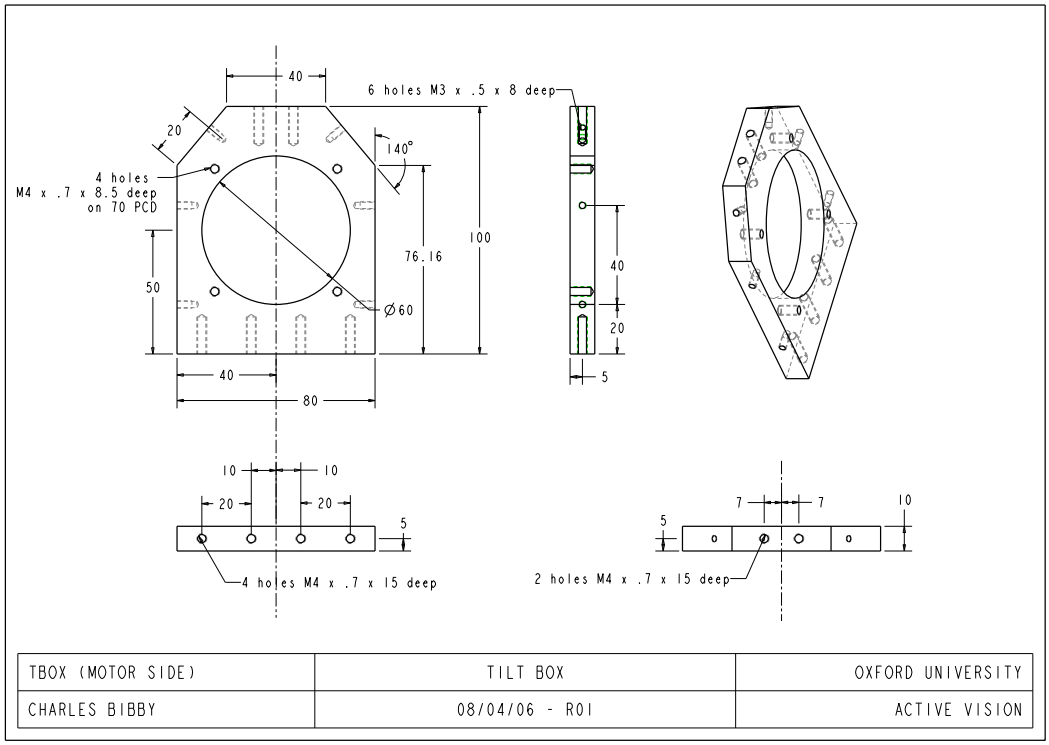
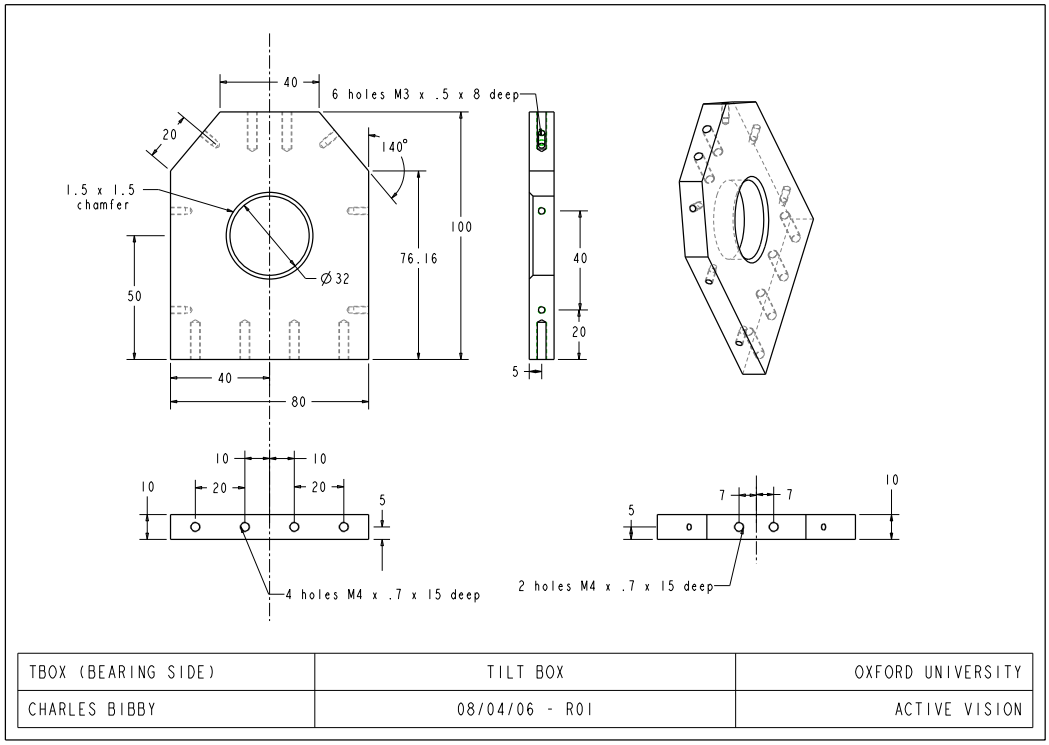


Figure E.7: Part – Tilt Box – Bottom.



TBOX (MOTOR SIDE)	TILT BOX	OXFORD UNIVERSITY
CHARLES BIBBY	08/04/06 - R01	ACTIVE VISION

Figure E.8: Part – Tilt Box – Motor Side.



TBOX (BEARING SIDE)	TILT BOX	OXFORD UNIVERSITY
CHARLES BIBBY	08/04/06 - R01	ACTIVE VISION

Figure E.9: Part – Tilt Box – Bearing Side.

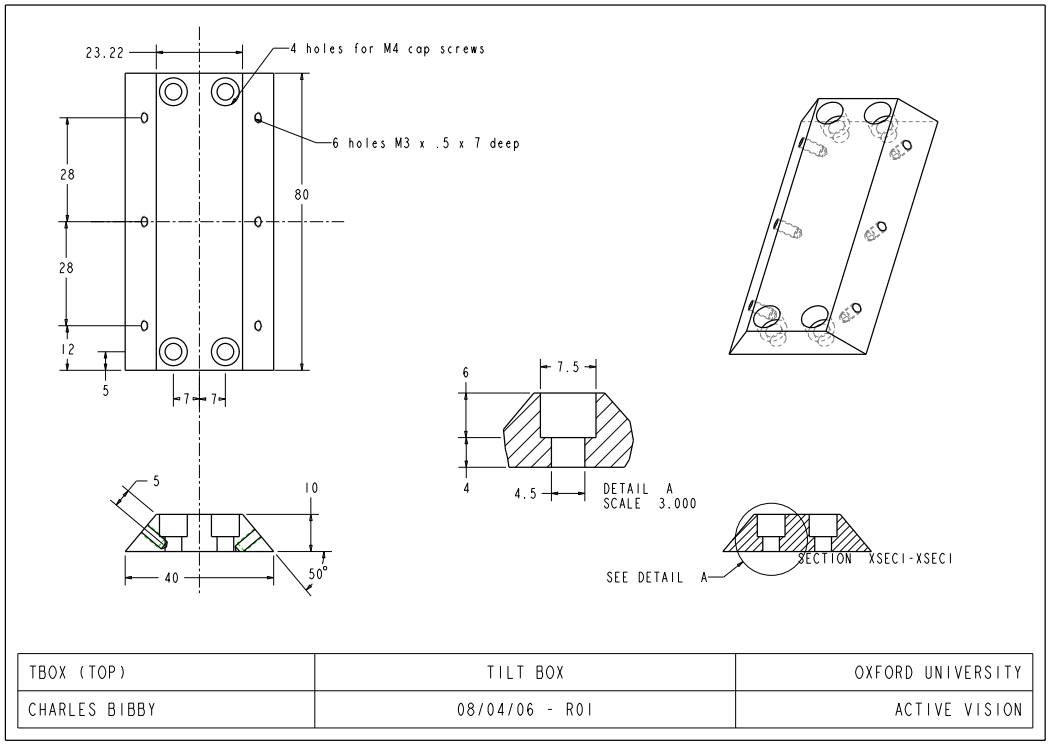


Figure E.10: Part – Tilt Box – Top.

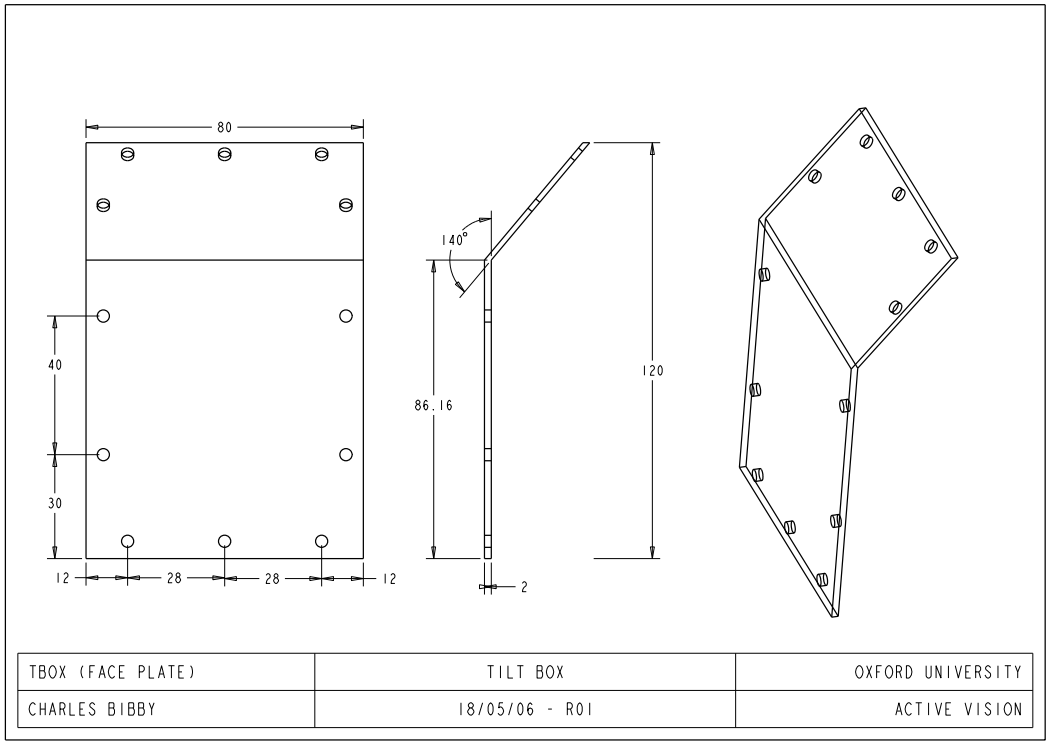


Figure E.11: Part – Tilt Box – Face Plate.

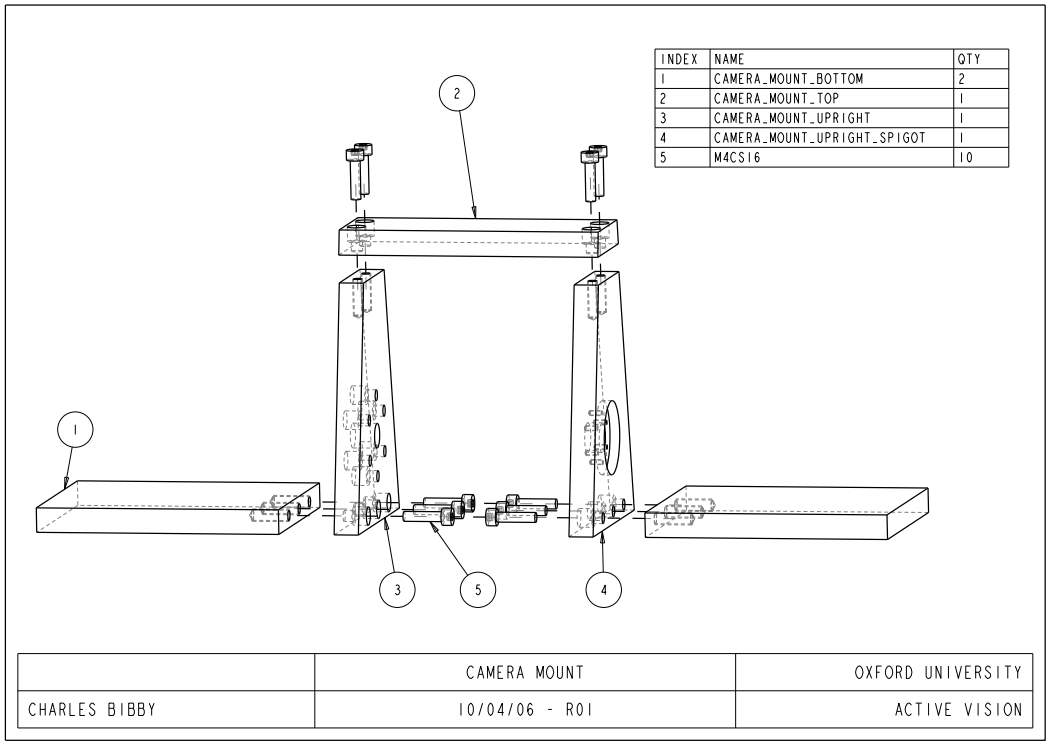


Figure E.12: Assembly – Camera Mount.

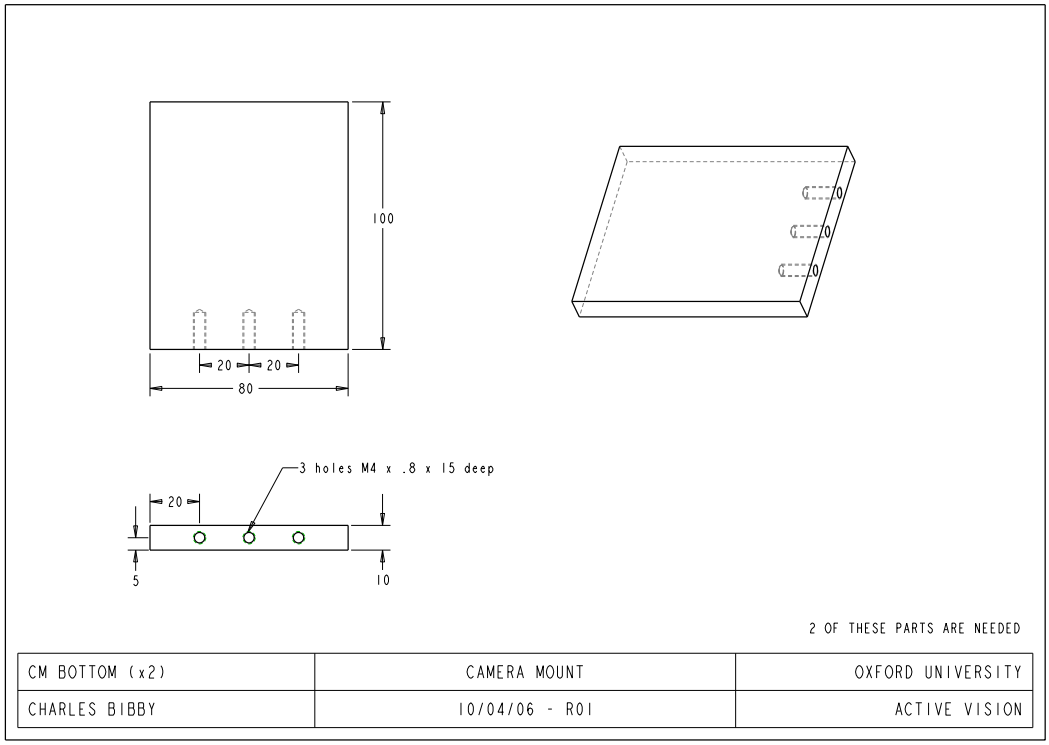


Figure E.13: Part – Camera Mount – Bottom.

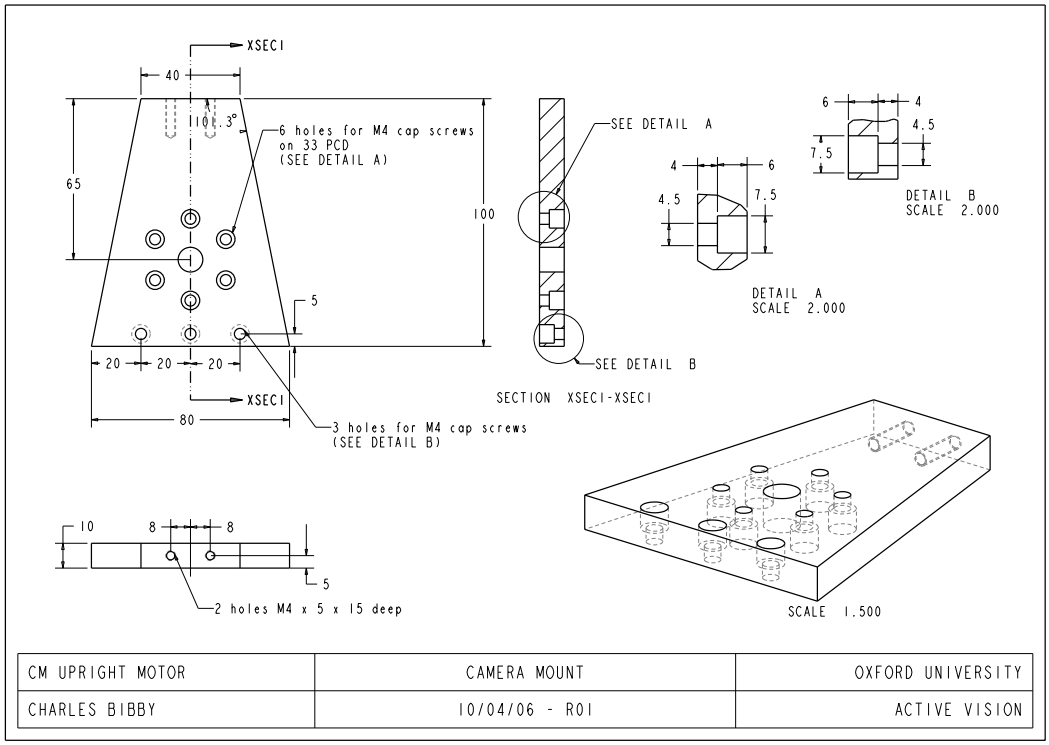


Figure E.14: Part – Camera Mount – Motor Side.

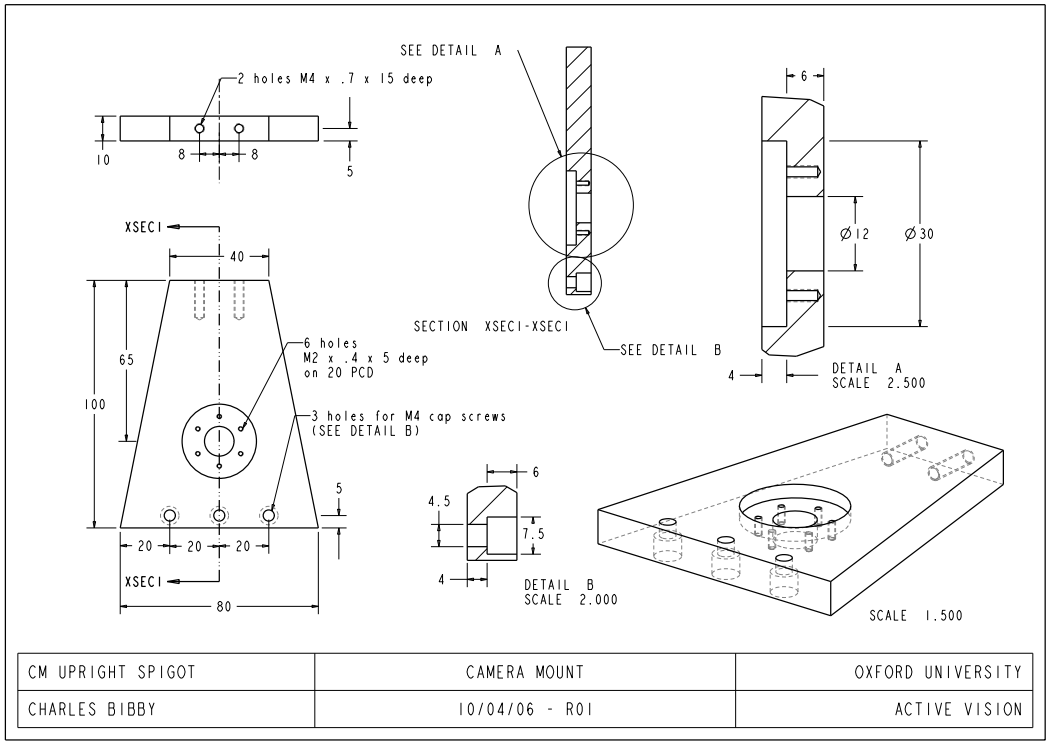


Figure E.15: Part – Camera Mount – Spigot Side.

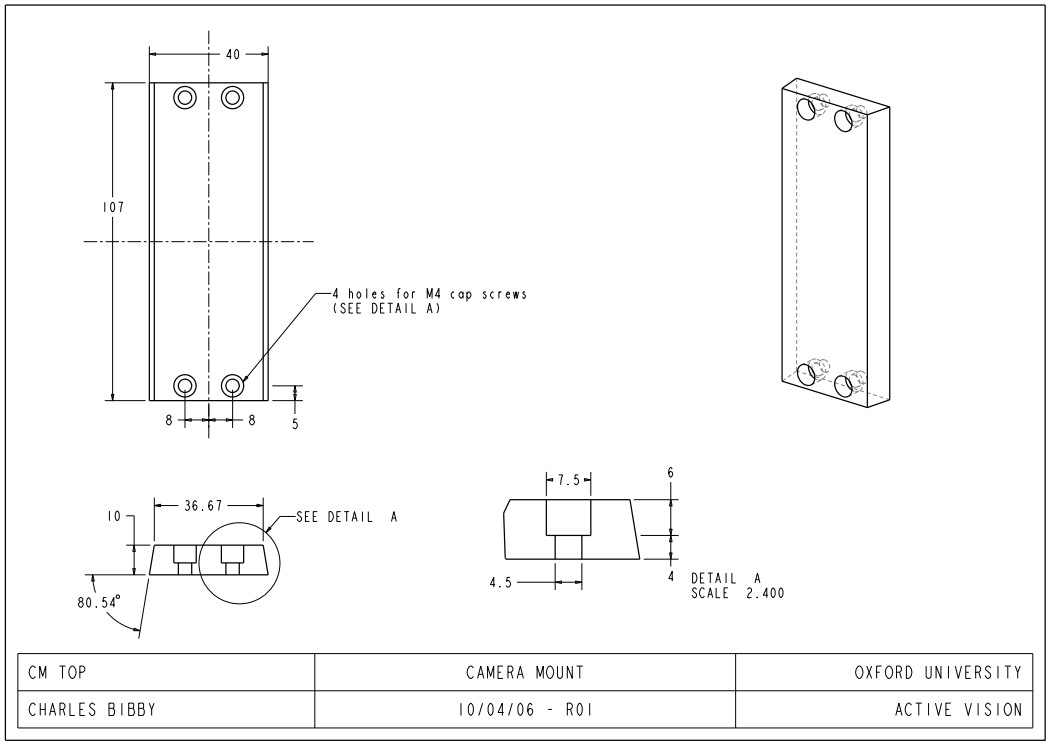


Figure E.16: Part – Camera Mount – Top.

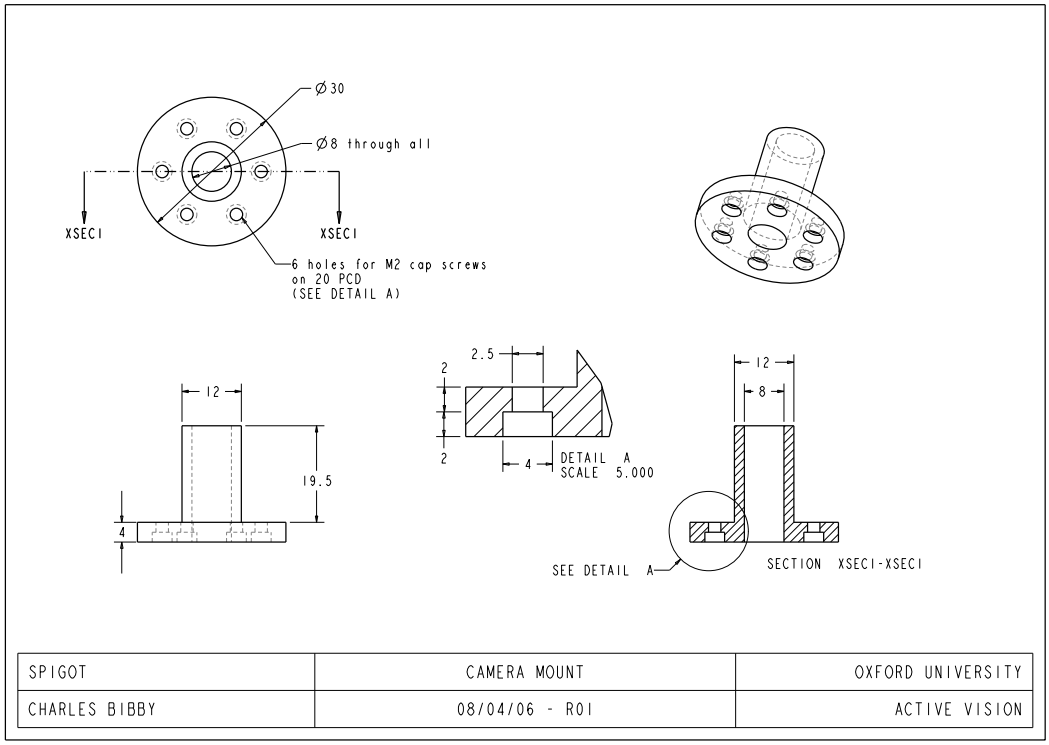


Figure E.17: Part – Spigot.

Appendix F

PTZ Design – Electrical

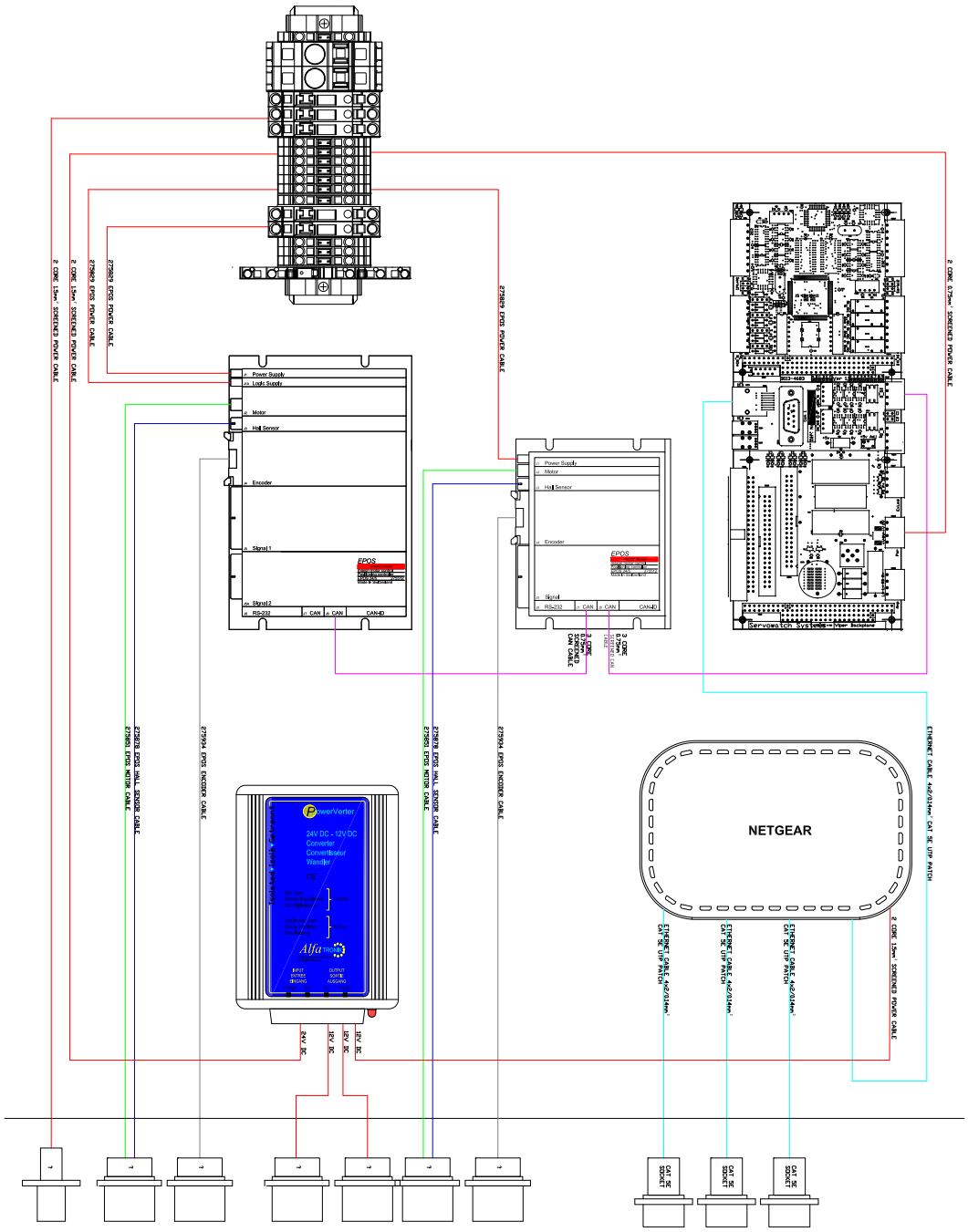


Figure F.1: Wiring – Embedded Controller.

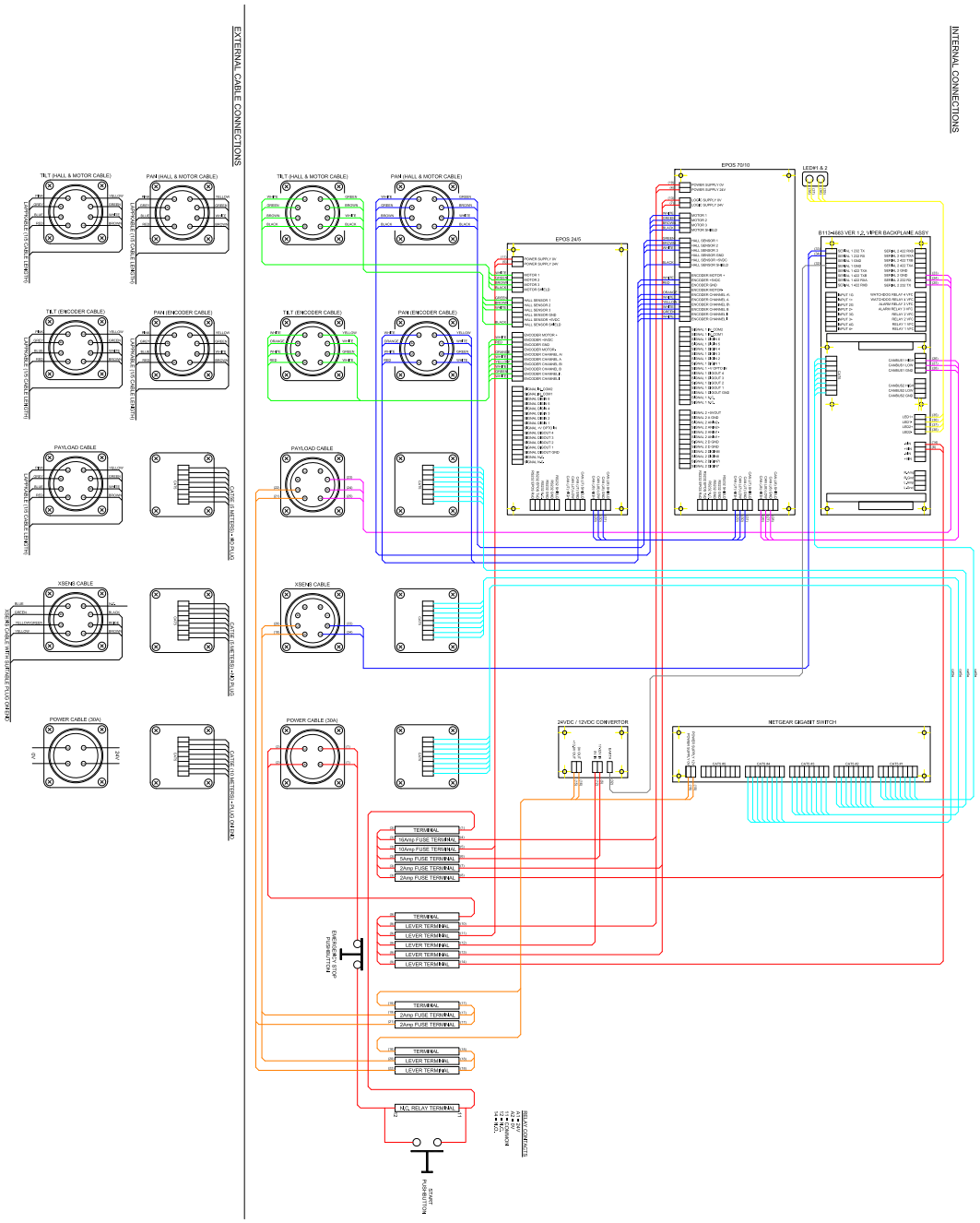


Figure F.2: Wiring – Embedded Controller – Detailed.

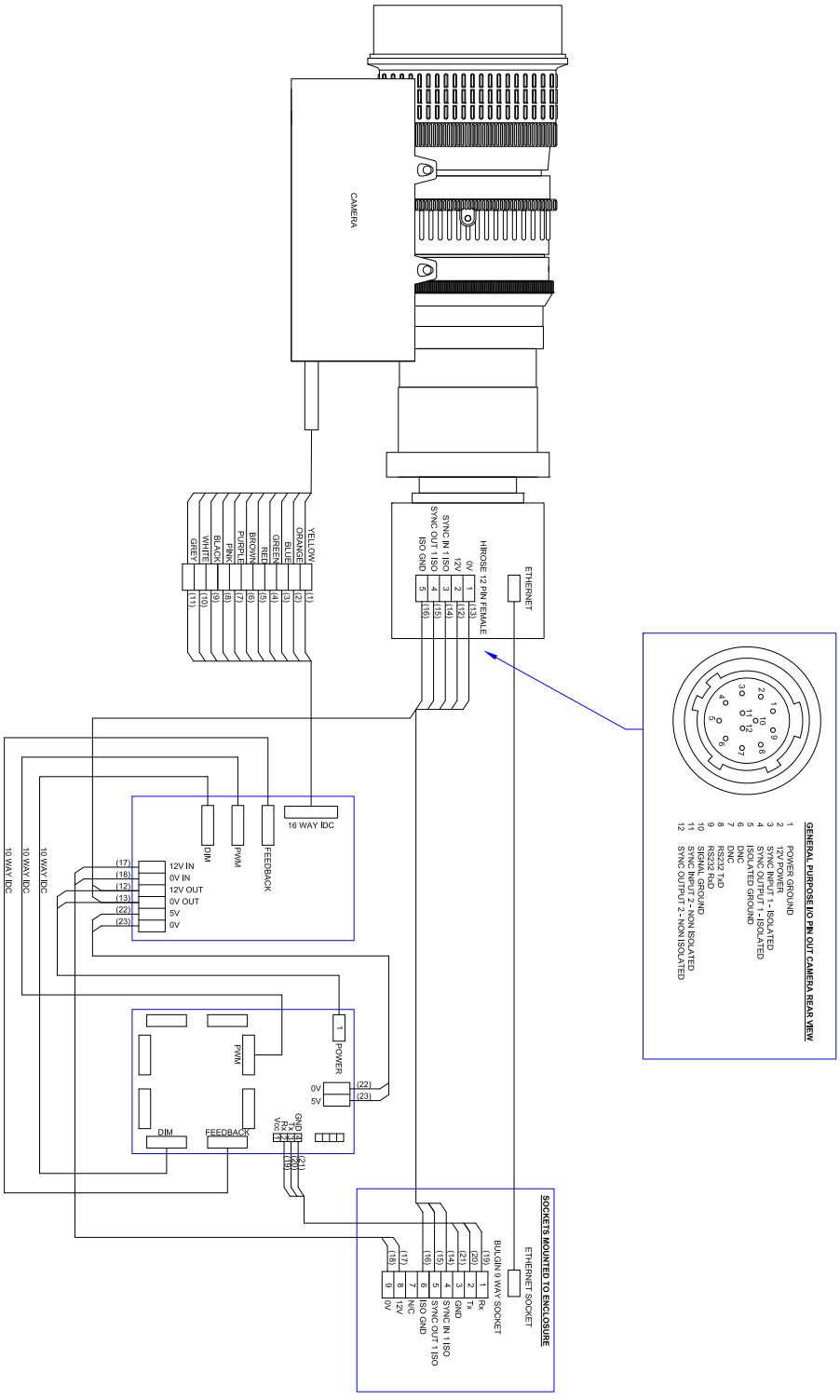


Figure F.3: Wiring – Payload.

Appendix G

Prototype System – Software

The software for the prototype system was broken down into separate processes. Each process would perform multiple tasks and communicate to other processes using interfaces, see Figure G.1. The interfaces used in the system were:

- **PTZInterface:** This carried commands to control the PTZ along with position, velocity and stabilisation feedback.
- **NavDataInterface:** This carried various forms of low-level navigation data from the GPS, IMU and AIS.
- **VisionInterface:** This carried the live video stream from the PTZ camera, managed which user had control of the PTZ and fed back any tracking information.
- **FusionInterface:** This fed back the hybrid map and object database, and allowed remote users to interact with the system.
- **WifiStabInterface:** This exchanged the vehicle's estimated location and heading, so that the steered antenna could point at the land based control centre. The current antenna direction was fed back and displayed on the hybrid map to show the user that the antenna was pointing correctly.
- **RConsoleInterface:** This gave remote access to low-level configuration, logging and timing information in any of the processes. It allowed an operator with access to any point on the network to maintain and monitor remote processes.

- **RJoystickInterface:** This gave remote control of a joystick, allowing one user at a time to control the PTZ. This user could be at any point on the network and during the demonstration was either on the sensor platform or in the shore based control centre.
- **OSIInterface:** This was the data exchange with the third party asset tracking software.
- **NetTesterInterface:** This was a utility to test network connectivity and performance under different types of load. This was used extensively in the development of the wireless telemetry link.

The processes on the system were:

- **pPTZServer:** This was compiled for an embedded Linux platform, it responded to remote requests and ran a 100Hz control loop to stabilise the PTZ.
- **pNavDataProcessor:** This collected navigation data from various serial interfaces and pushed the data onto the Ethernet for collection by multiple clients.
- **pWifiStab:** This took the vehicle's estimated location and heading and computed the required azimuth for the antenna, so that it would point back to the land based installation. The antenna steering hardware was controlled using a proprietary serial protocol.
- **pVision:** This performed the visual tracking presented in Chapters 6 and 7 and used this to do closed loop visual tracking with the PTZ. Other features of this software included the use of the OpenCV face detector [Viola & Jones, 2001] for automatic initialisation of faces and an experimental object database using random ferns [Ozuysal *et al.*, 2007] so objects could be recognised at a later date.
- **pFusion:** This used the methods presented in Chapters 4 and 5 to produce a hybrid map of the environment. It also generated an object database which included visual information received from pVision. This is where the PTZ was controlled from in the automatic and semi-automatic modes.
- **pVisionRemoteControl:** This allowed remote control of pVision and was used during development and testing.

- **pRFusion:** This gives a remote interface to pFusion and was used to display information at the land based control centre. Users are able to interact with the system and any requests are forwarded to pFusion.
- **pOSI:** This is a piece of third party software for asset control and tracking.
- **pNetworkTester:** This was used to test and benchmark network performance.
- **pRConsole:** This gave remote access to configuration, logging and timing information in any of the other processes.

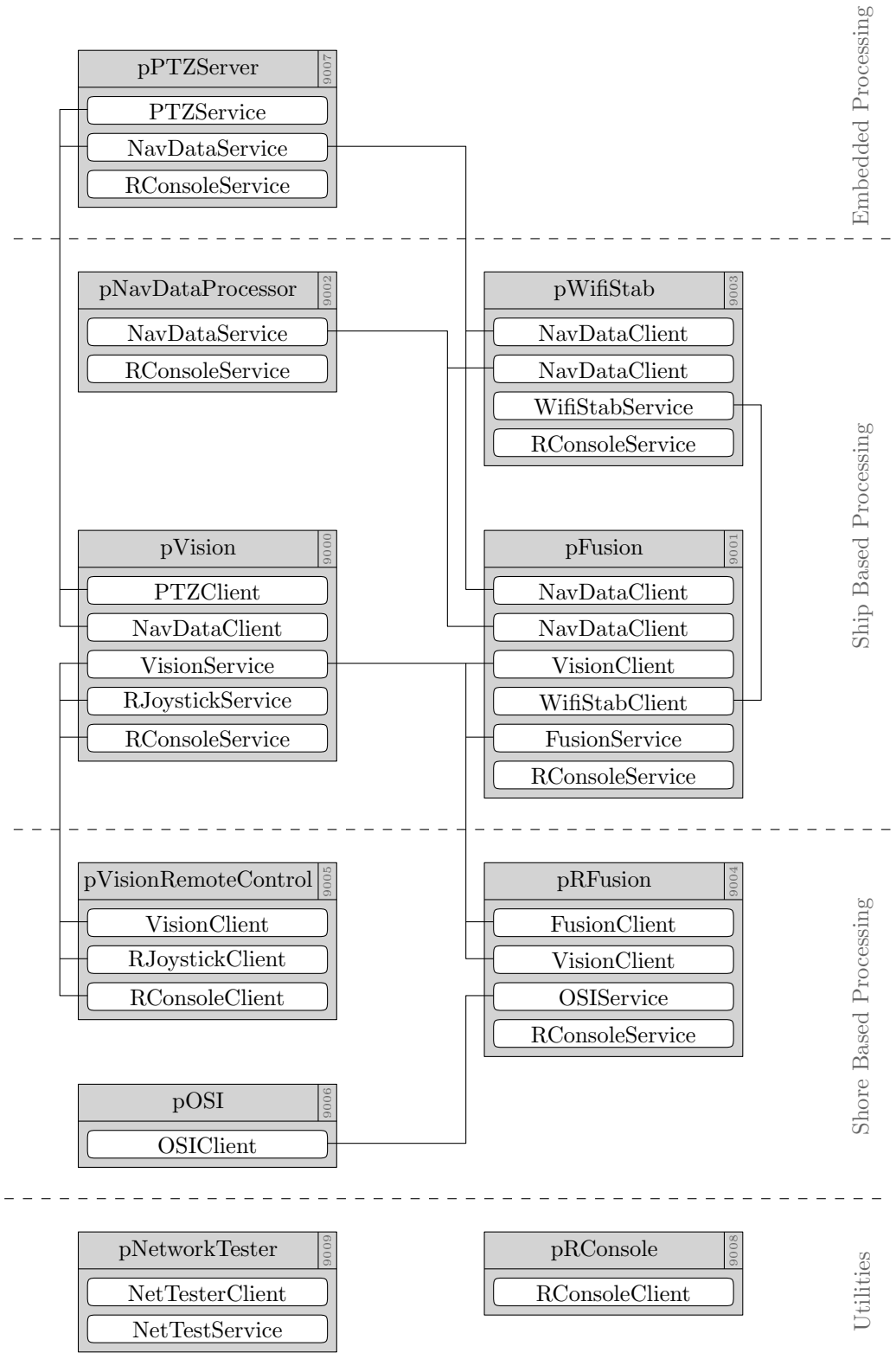


Figure G.1: System overview – software: (grey boxes) are processes and (white boxes) are interfaces.

Bibliography

Anderson, B.D.O., & Moore, J.B. 1979. *Optimal Filtering.* Prentice-Hall.

Baker, S., & Matthews, I. 2004. Lukas-kanade 20 years on: A unifiying framework. *Int'l Journal of Computer Vision*, **69**(3), 221–255.

Bar-Shalom, Y., & Fortmann, T. 1988. *Tracking and data association.* Academic Press.

Bar-Shalom, Y., Kirubarajan, T., & Li, X. R. 2002. *Estimation with Applications to Tracking and Navigation.* New York, NY, USA: John Wiley & Sons, Inc.

Besl, P. J., & Mckay, N. D. 1992. A Method for Registration of 3-D Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **14**, 239–256.

Bibby, C., & Reid, I. 2005. Visual Tracking at Sea. *In: Proc. IEEE Int'l Conf. on Robotics and Automation.*

Bibby, C., & Reid, I. 2006. Fast Feature Detection with a Graphics Processing Unit Implementation. *In: Proc. of the Int'l Workshop on Mobile Vision at the 9th European Conf. on Computer Vision, Graz, Austria.*

Bibby, C., & Reid, I. 2007. Simultaneous Localisation and Mapping in Dynamic Environments (SLAMIDE) with Reversible Data Association. *In: Proc. of Robotics Science and Systems.*

Bibby, C., & Reid, I. 2008. Robust Real-Time Visual Tracking using Pixel-Wise Posteriors. *In: Proc. 10th European Conf. on Computer Vision, Marseille, France.*

Bibby, C., & Reid, I. 2010a. A Hybrid SLAM Representation for Dynamic Marine Environments. *In: Proc. IEEE Int'l Conf. on Robotics and Automation.*

Bibby, C., & Reid, I. 2010b. Real-time Tracking of Multiple Occluding Objects using Level Sets. *In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Borenstein, J., Everett, H. R., & Feng, L. 1996. *Navigating Mobile Robots: Systems and Techniques*. A. K. Peters, Ltd., Wellesley, MA.

Burgard, W., Fox, D., Hennig, D., & Schmidt, T. 1996. Estimating the Absolute Position of a Mobile Robot Using Position Probability Grids. *Pages 896–901 of: Proc. 13th National Conf. on Artificial Intelligence, Menlo Park*.

Burgard, W., Fox, D., Jans, H., Matenar, C., & Thrun, S. 1999. Sonar-Based Mapping of Large-Scale Mobile Robot Environments using EM. *Pages 67–76 of: Proc. 16th Int'l Conf. on Machine Learning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Castellanos, J.A., Neira, J., & Tardos, J.D. 2004. Limits to the consistency of EKF-based SLAM. *In: Proc. Int'l Joint Conf. on Artificial Intelligence, Vancouver, Canada*.

Celeux, G., & Govaert, G. 1992. A Classification EM algorithm for clustering and two stochastic versions. *Computational Statistics and Data Analysis*, **14**(3), 315–332.

Chan, T., & Vese, L. 2001. Active contours without edges. *IEEE Trans. on Image Processing*, **10**(2), 266–277.

Collins, R. T. 2003. Mean-shift Blob Tracking through Scale Space. *Pages 234–240 of: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*, vol. 2.

Comaniciu, D., Ramesh, V., & Meer, P. 2000. Real-time tracking of non-rigid objects using mean shift. *Pages 142–149 of: Proc. 19th IEEE Conf. on Computer Vision and Pattern Recognition, Hilton Head Island*, vol. 2.

Cremers, D. 2006. Dynamical statistical shape priors for level set based tracking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **28**(8), 1262–1273.

Cremers, D., Rousson, M., & Deriche, R. 2007. A Review of Statistical Approaches to Level Set Segmentation: Integrating Color, Texture, Motion and Shape. *Int'l Journal of Computer Vision*, **V72**(2), 195–215.

Csorba, M. 1997. *Simultaneous Localisation and Map Building*. Ph.D. thesis, Oxford University.

Davis, T. 2006. *Direct Methods for Sparse Linear Systems*. SIAM.

Dellaert, F. 2002. *The Expectation Maximization Algorithm*. Tech. rept. GIT-GVU-02-20. Georgia Institute of Technology.

Dellaert, F., & Kaess, M. 2006. Square Root SAM. *Int'l Journal of Robotics Research*.

Dellaert, F., Fox, D., Burgard, W., & Thrun, S. 1999. Monte Carlo Localization for Mobile Robots. *In: Proc. IEEE Int'l Conf. on Robotics and Automation*.

Dempster, A. P., Laird, N. M., & Rubin, D. B. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society*.

Dissanayake, G., Williams, S. B., Durrant-whyte, H., & Bailey, T. 2002. Map Management for Efficient Simultaneous Localization and Mapping (SLAM). *Autonomous Robots*, **12**, 267–286.

Dissanayake, M.W.M.G., Newman, P., Clark, S., Durrant-Whyte, H.F., & Csorba., M. 2001. A solution to the simultaneous localization and map building (SLAM) problem. *IEEE Trans. Robotics and Automation*, **17**, 229–241.

Drummond, T. W., & Cipolla, R. 2000. Real-time tracking of multiple articulated structures in multiple views. *In: Proc. 6th European Conf. on Computer Vision, Dublin*.

Elfes, A. 1987. Sonar-based real-world mapping and navigation. *IEEE Trans. Robotics and Automation*, **3**, 249–265.

Elfes, A. 1989. Using Occupancy Grids for Mobile Robot Perception and Navigation. *Computer*, **22**(6), 46–57.

Eustice, R., Singh, H., Leonard, J., Walter, M., & Ballard, R. 2005. Visually Navigating the RMS Titanic with SLAM Information Filters. *In: Proc. of Robotics Science and Systems*.

Evans, L. C. 2002. *Partial Differential Equations.* AMS.

Fisher, R.B. 2004. *CAVIAR Test Case Scenarios, EC Funded IST 2001 37540.* Online Book.

Fox, D. 2001. KLD-Sampling: Adaptive Particle Filters. *Pages 713–720 of: Proc. Conf. on Neural Information Processing Systems.*

Freedman, D., & Zhang, T. 2004. Active contours for tracking distributions. *IEEE Trans. on Image Processing*, **13**(4), 518–526.

Goldenberg, R., Kimmel, R., Rivlin, E., & Rudzsky, M. 2001. Fast Geodesic Active Contours. *IEEE Trans. on Image Processing*, **10(10)**, 1467–75.

Grimson, W. E. L. 1990. *Object Recognition by Computer: The Role of Geometric Constraints.* Cambridge MA: MIT Press.

Guivant, J., & Nebot, E. 2001. Optimization of the Simultaneous Localization and Map Building Algorithm for Real Time Implementation. *IEEE Trans. Robotics and Automation*, **17**, 242–257.

Hähnel, D., Triebel, R., Burgard, W., & Thrun, S. 2003. Map Building with Mobile Robots in Dynamic Environments. *In: Proc. IEEE Int'l Conf. on Robotics and Automation.*

Harris, C. 1993. Tracking with rigid models. *In: Active vision.* MIT Press.

Hartley, R., & Zisserman, A. 2004. *Multiple View Geometry in Computer Vision.* Cambridge University Press.

Isard, M., & Blake, A. 1998a. CONDENSATION - conditional density propagation for visual tracking. *Int'l Journal of Computer Vision*, **29**, 5–28.

Isard, M., & Blake, A. 1998b. ICondensation: Unifying low-level and high-level tracking in a stochastic framework. *Pages 893–908 of: Proc. 5th European Conf. on Computer Vision, Freiburg.*

Isard, M., & MacCormick, J. 2001. BraMBLe: A Bayesian multiple-blob tracker. *Pages 34–41 of: Proc. 8th Int'l Conf. on Computer Vision, Vancouver.*

Jebara, T. 2003. Images as Bags of Pixels. *In: Proc. 9th Int'l Conf. on Computer Vision, Nice.*

Jensfelt, P., & Kristensen, S. 1999. Active Global Localisation for a Mobile Robot Using Multiple Hypothesis Tracking. *Pages 13–22 of: IEEE Trans. Robotics and Automation.*

Jepson, A. D., Fleet, D. J., & Black, M. J. 2002. A Layered Motion Representation with Occlusion and Compact Spatial Support. *In: Proc. 7th European Conf. on Computer Vision, Copenhagen.*

Jojic, N., & Frey, B. 2001. Learning flexible sprites in video layers. *In: Proc. 20th IEEE Conf. on Computer Vision and Pattern Recognition, Hawaii.*

Kadir, T., & Brady, M. 2005. Estimating statistics in arbitrary regions of interest. *In: Proc. 16th British Machine Vision Conf., Oxford.*

Kalman, R. E. 1960. A New Approach to Linear Filtering and Prediction Problems. *Trans. of the ASME–Journal of Basic Engineering*, **82**(Series D), 35–45.

Kass, M., Witkin, A., & Terzopoulos, D. 1988. Snakes: Active contour models. *Int'l Journal of Computer Vision*, **1**, 321–331.

Khan, Z., Balch, T., & Dellaert, F. 2004. An MCMC-based particle filter for tracking multiple interacting targets. *Pages 279–290 of: Proc. 8th European Conf. on Computer Vision, Prague.*

Leonard, J., & Rikoski, R. 2001. Incorporation of delayed decision making into stochastic mapping. *In: Experimental Robotics VII, Lecture Notes in Control and Information Sciences.*

Leonard, J.J., & Durrant-Whyte, H.F. 1991. Mobile robot localization by tracking geometric beacons. *IEEE Trans. Robotics and Automation*, **7**, 376–382.

Leonard, J.J., & Feder, H.J.S. 1999. A computationally efficient method for large-scale concurrent mapping and localization. *In: Proc. 9th Int'l Symposium on Robotics Research.*

Leonard, J.J., & Feder, H.J.S. 2001. Decoupled stochastic mapping. *IEEE Journal of Oceanic Engineering*, **26**, 561 – 571.

Li, C., Xu, C., Gui, C., & Fox, M. D. 2005. Level Set Evolution without Re-Initialization: A New Variational Formulation. *Pages 430–436 of: Proc. 22nd IEEE Conf. on Computer Vision and Pattern Recognition, San Diego, California*, vol. 1. IEEE Computer Society.

Lowe, D. 1999. Object Recognition from Local Scale-Invariant Features. *In: Proc. Int'l Conf. on Computer Vision.*

Lucas, B.D., & Kanade, T. 1981. An Iterative Image Registration Technique with an Application to Stereo Vision. *In: Proc. Int'l Joint Conf. on Artificial Intelligence.*

MacCormick, J., & Blake, A. 1999. A probabilistic exclusion principle for tracking multiple objects. *Pages 572–587 of: Proc. 7th Int'l Conf. on Computer Vision, Corfu.*

Manyika, J., & Durrant-Whyte, H. 1994. *Data Fusion and Sensor Management a decentralized information-theoretic approach.* Ellis Horwood.

Meila, M., & Heckerman, D. 1998. An Experimental Comparison of Several Clustering and Initialization Methods. *In: Proc. 14th Conf. on Uncertainty in Artificial Intelligence.*

Moeslund, T.B., Hilton, A., & Krüger, V. 2006. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, **104**, 90–126.

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit., B. 2002. FastSLAM: A Factored Solution to the Simultaneous Localization and Mapping Problem. *In: Proc. Int'l Joint Conf. on Artificial Intelligence.*

Montemerlo, M., Thrun, S., Koller, D., & Wegbreit, B. 2003. FastSLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Converges. *Pages 1151–1156 of: Proc. Int'l Joint Conf. on Artificial Intelligence.*

Moravec, H. 1988. Sensor fusion in certainty grids for mobile robots. *Artificial Intelligence Magazine*, **9**, 61–74.

Moravec, H., & Elfes, A. E. 1985. High Resolution Maps from Wide Angle Sonar. *Pages 116 – 121 of: Proc. IEEE Int'l Conf. on Robotics and Automation.*

Murray, D. W., Du, F., McLauchlan, P. F., Reid, I. D., Sharkey, P. M., & Brady, J. M. 1992. Design of Stereo Heads. *In: Active Vision.* MIT Press.

Neal, R., & Hinton, G. 1998. A View of the EM Algorithm that Justifies Incremental, Sparse, and other Variants. *In: Learning in Graphical Models.* Kluwer.

Neira, J., & Tardos, J. 2001. Data association in stochastic mapping using the joint compatibility test. *In: IEEE Trans. Robotics and Automation.*

Nettleton, E., Thrun, S., & Durrant-Whyte, H. 2003. Decentralised SLAM with Low-Bandwidth Communication for Teams of Airborne Vehicles. *In: Proc. Int'l Conf. on Field and Service Robotics.*

Newman, P.M., & Durrant-Whyte, H.F. 2001. A new solution to the simultaneous and map building (SLAM) problem. *In: Proc. SPIE.*

Nieto, J., Guivant, J., & Nebot, E. 2004. The hybrid metric maps (HYMMS): A novel map representation for DenseSLAM. *Pages 391–396 of: Proc. IEEE Int'l Conf. on Robotics and Automation.*

Nitzberg, M., & Mumford, D. 1990. The 2.1-D Sketch. *In: Proc. 3rd Int'l Conf. on Computer Vision, Osaka.*

Osher, S., & Paragios, N. 2003. *Geometric Level Set Methods in Imaging, Vision and Graphics.* Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Ozuysal, M., Fua, P., & Lepetit, V. 2007. Fast Keypoint Recognition in Ten Lines of Code. *In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition.*

Pandey, A. K., Krishna, K. M., & Nath, M. 2007. Feature Based Occupancy Grid Maps for Sonar Based Safe-Mapping. *In: Proc. Int'l Joint Conf. on Artificial Intelligence.*

Paragios, N., & Deriche, R. 2000. Geodesic Active Contours and Level Sets for the Detection and Tracking of Moving Objects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **22**(3), 266–280.

Paskin, M. A. 2003. Thin Junction Tree Filters for Simultaneous Localization and Mapping. *Pages 1157–1164 of: Proc. Int'l Joint Conf. on Artificial Intelligence.*

Pedraza, L., Dissanayake, G., Miro, J. Valls, Rodriguez-Losada, D., & Matia, F. 2007. BS-SLAM: Shaping the World. *In: Proc. of Robotics Science and Systems.*

Reece, S., & Roberts, S. 2005. Robust, Low-Bandwidth, Multi-Vehicle Mapping. *In: Proc. Int'l Conf. on Information Fusion.*

Reid, D.B. 1979. An Algorithm for Tracking Multiple Targets. *IEEE Trans. on Automatic Control*, **24**, 843–854.

Reid, I.D., & Connor, K. 2005. Multiview Segmentation and Tracking of Dynamic Occluding Layers. *Pages 919–928 of: Proc. 16th British Machine Vision Conf., Oxford*, vol. 2.

Rother, C., Kolmogorov, V., & Blake, A. 2004. GrabCut: Interactive Foreground Extraction using Iterated Graph Cuts. *In: ACM Transactions on Graphics.*

Roumeliotis, S.I., & Bekey, G.A. 2000. Bayesian estimation and Kalman filtering: a unified framework formobile robot localization. *In: Proc. IEEE Int'l Conf. on Robotics and Automation.*

Scharf, L. L., & McWhorter, L. T. 1993. Geometry of the Cramer-Rao bound. *Signal Processing*, **31**(3), 301–311.

Schmaltz, C., Rosenhahn, B., Brox, T., Cremers, D., Weickert, J., Wietzke, L., & Sommer, G. 2007. Occlusion Modeling by Tracking Multiple Objects. *In: Pattern Recognition.*

Sibley, G., Sukhatme, G. S., & Matthies, L. 2007. Constant Time Sliding Window Filter SLAM as a Basis for Metric Visual Perception. *In: Proc. IEEE Int'l Conf. on Robotics and Automation.*

Simmons, R., & Koenig, S. 1995. Probabilistic Robot Navigation in Partially Observable Environments. *Pages 1080–1087 of: Proc. Int'l Joint Conf. on Artificial Intelligence.*

Smith, R., Self, M., & Cheeseman, P. 1988. A stochastic map for uncertain spatial relationships. *Pages 467–474 of: Proc. 4th Int'l Symposium on Robotics Research.* MIT Press.

Sommerlade, E., & Reid, I. 2008. Information theoretic Active Scene Exploration. *In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition.*

Soto, C., Song, B., & Roy-Chowdhury, A. 2009. Distributed Multi-Target Tracking In A Self-Configuring Camera Network. *In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition.*

Tao, H., Sawhney, H. S., & Kumar, R. 2000. Dynamic Layer Representation with Applications to Tracking. *In: Proc. 19th IEEE Conf. on Computer Vision and Pattern Recognition, Hilton Head Island.*

Tardos, J. D., Neira, J., Newman, P. M., & Leonard, J. J. 2002. Robust mapping and localization in indoor environments using sonar data. *Int'l Journal of Robotics Research*, **21**, 2002.

Thrun, S. 2002. Learning Occupancy Grids With Forward Sensor Models. *Autonomous Robots*, **15**, 111–127.

Thrun, S., Burgard, W., & Fox, D. 1998. A probabilistic approach to concurrent mapping and localization for mobile robots. *Pages 29–53 of: Machine Learning*.

Thrun, S., Liu, Y., Koller, D., Ng, A.Y., Ghahramani, Z., & Durrant-Whyte, H. 2004. Simultaneous Localization and Mapping with Sparse Extended Information Filters. *Int'l Journal of Robotics Research*, **23**, 693–716.

Thrun, S., Burgard, W., & Fox, D. 2005. *Probabilistic Robotics*. MIT Press.

Triggs, B., McLauchlan, P., Hartley, R., & Fitzgibbon, A. 2000. Bundle Adjustment – A Modern Synthesis. *Pages 298–372 of: Vision Algorithms: Theory and Practice*. Lecture Notes in Computer Science, vol. 1883. Springer-Verlag.

Vermaak, J., & Doucet, A. 2003. Maintaining multi-modality through mixture tracking. *Pages 1110–1116 of: Proc. 9th Int'l Conf. on Computer Vision, Nice*.

Vese, L. A., & Chan, T. F. 2002. A multiphase level set framework for image segmentation using the Mumford and Shah model. *Int'l Journal of Computer Vision*, **50**, 271–293.

Viola, P., & Jones, M. 2001. Robust Real-time Object Detection. *In: Int'l Journal of Computer Vision*.

Walter, M., & Leonard, J. 2004. An Experimental Investigation of Cooperative SLAM. *In: Proc. 5th IFAC Symposium on Intelligent Autonomous Vehicles*.

Wang, C. C., Thorpe, C., & Thrun, S. 2003. Online Simultaneous Localization And Mapping with Detection And Tracking of Moving Objects: Theory and Results from a Ground Vehicle in Crowded Urban Areas. *In: Proc. IEEE Int'l Conf. on Robotics and Automation*.

Wang, J.Y.A., & Adelson, E.H. 1993. Layered representation for motion analysis. *Pages 361–366 of: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition*.

Wijesoma, W. S., Perera, L. D. L., & Adams, M. D. 2006. Toward Multidimensional Assignment Data Association in Robot Localization and Mapping. *In: IEEE Trans. Robotics*.

Williams, S.B. 2001. *Efficient Solutions to Autonomous Mapping and Navigation Problems*. Ph.D. thesis, University of Sydney.

Wolf, D., & Sukhatme, G. S. 2004. Online Simultaneous Localization And Mapping in Dynamic Environments. *In: Proc. IEEE Int'l Conf. on Robotics and Automation*.

Yilmaz, A. 2007. Object Tracking by Asymmetric Kernel Mean Shift with Automatic Scale and Orientation Selection. *In: Proc. IEEE Conf. on Computer Vision and Pattern Recognition, Minneapolis, Minnesota*.

Yilmaz, A., Javed, O., & Shah, M. 2006. Object tracking: A survey. *ACM Computing Surveys*, **38**.

Zhang, T., & Freedman, D. 2005. Improving performance of distribution tracking through background matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(2), 282–287.