Extensions to the Probabilistic Multi-Hypothesis Tracker for Tracking, Navigation and SLAM

Brian Cheung

Thesis submitted for the degree of

Doctor of Philosophy

School of Electrical and Electronic Engineering
Faculty of Engineering
The University of Adelaide
Adelaide, South Australia

February 2012

Contents

1 Introduction **1**

1.1 Contributions 3

1.2 Thesis Overview 3

2 Background **7**

2.1 Data Association 7

2.1.1 Nearest Neighbour 8

2.1.2 Probabilistic Data Association Filter 8

2.1.3 Probabilistic Multi-Hypothesis Tracker 9

2.2 State Estimation 10

2.2.1 Problem Definition 11

2.2.2 Target Dynamic Models 11

2.2.3 Measurement Model 14

2.2.4 Kalman Filter 15

2.2.5 Extended Kalman Filter 16

2.2.6 Unscented Kalman Filter 17

2.2.7 Particle Filter 19

2.3 Probabilistic Multi-Hypothesis Tracker 21

2.3.1 Problem Definition 21

2.3.2 PMHT for multi-target tracking 22

3 PMHT with time uncertainty **27**

3.1 Introduction 27

3.2 Problem Formulation 28

3.3 PMHT for tracking with timing uncertainty 29

3.4 Performance Analysis 34
3.5 Comparison of PMHT-t with other Methods 41
3.5.1 PMHT-t Solution 44
3.5.2 Sliding Window PMHT-t 46
3.5.3 Alternative Algorithms 48
3.5.4 Performance Analysis 49
3.6 Conclusion 51

4 PMHT Path Planning **53**
4.1 Introduction 53
4.1.1 Background 54
4.1.2 Proposed Approach 56
4.2 Problem Formulation 57
4.3 Path planning problem 59
4.4 PMHT for multiple platform path planning 61
4.5 Simulation Results 65
4.6 Tradeoff between smoothness and proximity 71
4.7 Locale Density Dependence 73
4.8 Genetic Algorithm Solution to the Travelling Salesmen Problem 77
4.8.1 GA-TSP with PMHT Smoother 79
4.8.2 Path Planning Comparison 79
4.9 Sliding batch PMHT Path Planning 88
4.10 PMHT-pp for Indoor Environments 91
4.10.1 PMHT-pp-pf Indoor Strategies 91
4.10.2 PMHT-pp-pf Indoor Results 92
4.11 PMHT-pp with Non-Homogeneous Locales 98
4.11.1 Non-Homogeneous Locales 98
4.11.2 PMHT-pp with Priority as a Continuous Density 99
4.11.3 Simulation Example 102
4.12 Conclusion 105

5 PMHT-c for SLAM **107**
5.1 Introduction 107

5.2 Problem Formulation 110
5.3 SLAM formulation 111
5.4 The PMHT with Classification in SLAM 114
5.5 Performance Analysis 117
5.5.1 Simulated results 117
5.5.2 Victoria Park Data 123
5.6 Conclusions 127

6 Conclusion **129**
6.1 Tracking with Time Uncertainty 129
6.2 Multiple Platform Path Planning 130
6.3 SLAM with classifications 131
6.4 Future Work 132

A Source Code Listing **133**
A.1 PMHT-t Source Code 133
A.2 PMHT-pp Source Code 136
A.3 PMHT-c SLAM Source Code 141

List of Figures

3.1 Time error pmf 35
3.2 PMHT-t Scenario 1 Tracking Results 37
3.3 PMHT-t Scenario 2 Tracking Results 39
3.4 PMHT-t Scenario 2 with Clutter Tracking Results 42

4.1 Locale Example Map 58
4.2 PMHT-pp assigned trajectories evolution for 4 platforms 67
4.3 Assorted PMHT-pp Planned Trajectories 68
4.4 Converged estimates of Π^k 70
4.5 Converged estimates of Π^τ 70
4.6 PMHT-pp Tradeoff using ratio between Q and R 72
4.7 PMHT-pp with varied grid of locales 74
4.8 PMHT-pp with varied random locales 76
4.9 GA-TSP with a grid of locales 78
4.10 Initial priors for PMHT-pp smoothing 80
4.11 GA-TSP with PMHT-pp smoothed output 81
4.12 Results for 4 platforms and grid of locales 83
4.13 Results for 3 platforms and random locales 84
4.14 Results for 4 platforms and random locales 85
4.15 Locale distribution between platforms 87
4.16 Trajectories separated for the four platforms after 8 iterations 89
4.17 Trajectories at each iteration 90
4.18 PMHT-pp-pf with grid of locales 94
4.19 Comparison between PMHT-pp and PMHT-pp-pf 95
4.20 PMHT-pp-pf with 4 walls 96

4.21 PMHT-pp-pf in an indoor environment 97
4.22 PMHT-pp with Priority Map Comparison 104

5.1 Example platform trajectory with random landmarks. 119
5.2 Percentage of divergent tracks comparison 120
5.3 RMS position estimation error comparison 121
5.4 Percentage of divergent tracks with PMHT comparison 122
5.5 RMS position estimation error with PMHT comparison 122
5.6 Divergent tracks with different misclassified measurements 123
5.7 Divergent tracks with mismatched misclassified measurements 124
5.8 Histogram of tree widths 125
5.9 PMHT-c estimated trajectory. 126

List of Tables

3.1 PMHT-t Scenario 1 Monte Carlo RMS results 38
3.2 PMHT-t Scenario 2 Monte Carlo RMS results 40
3.3 PMHT-t Scenario 2 with Clutter Monte Carlo RMS results 43
3.4 Monte Carlo Position RMS Comparison results 50
3.5 Monte Carlo Timing Error Mean RMS Comparison results 51
3.6 Monte Carlo Timing Error Precision RMS Comparison results 51

4.1 PMHT-pp Monte Carlo results for grid of locales 75
4.2 PMHT-pp Monte Carlo results for random locales 75
4.3 Path Planning Monte Carlo Comparison results 86

Abstract

Multi-target tracking is a problem that involves estimating target states from noisy data whilst simultaneously deciding which measurement was produced by each target. The Probabilistic Multi-Hypothesis Tracker (PMHT) is an algorithm that solves the multi-target tracking problem. This thesis presents extensions to the PMHT to address problems that may arise in the use of real sensors and considers multi-target tracking techniques for use in other applications such as autonomous vehicles.

It is generally assumed that a sensor collects a set of noisy position measurements at known times. In some situations, the time information may not be reliable and cause filtering issues. This thesis derives an extension to the PMHT that introduces an assignment index that identifies the true time at which a measurement was collected. This extension of the PMHT allows for tracking on measurements with time errors, such as time delays. A further extension allows the PMHT algorithm to simultaneously estimate the time error parameters whilst tracking targets.

The above extension is applied to the problem of planning paths for multiple platforms to explore an unknown area. Given a set of locales to be visited and the platform initial positions, the path planning problem has the same mathematical form as a multi-target tracking problem, with locales as measurements and the platforms as targets. The extended PMHT algorithm uses hypothesised time-stamps to associate locales to platforms and times simultaneously.

Autonomous vehicles are expected to use information from their sensors to navigate and map their environment. Simultaneous localisation and mapping (SLAM) is the name given to this task and is essentially a multi-target tracking problem. This thesis proposes the use of PMHT and landmark classification information received with measurements to improve the performance of SLAM.

x

Declaration

I, Brian Cheung certify that this work contains no material which has been accepted for the award of any other degree or diploma in any university or other tertiary institution and, to the best of my knowledge and belief, contains no material previously published or written by another person, except where due reference has been made in the text.

I give my consent to this copy of my thesis, when deposited in the University Library, being made available for loan and photocopying, subject to the provisions of the Copyright Act 1968.

The author acknowledges that copyright of published works contained within this thesis (as listed below*) resides with the copyright holder(s) of those works.

I also give permission for the digital version of my thesis to be made available on the web, via the University's digital research repository, the Library catalogue and also through web search engines, unless permission has been granted by the University to restrict access for a period of time.

Signature

Date

Acknowledgements

There are many people who have helped me along the way and to whom I would like to thank you.

To my supervisor at the Defence Science and Technology Organisation, Samuel Davey. Thanks for your open door and guidance every time I was lost. Thanks for being open to my ideas and stepping through ideas with me.

Thanks to Neil Gordon, the head of Tracking and Sensor Fusion group in ISRD. For encouraging me to start a PhD at the beginning and for pushing me to hurry up and finish at the end!

To Professor Doug Gray, of the School of Electrical and Electronic Engineering at the University of Adelaide, for your constructive guidance, demanding a high standard of research and for making me think like a scientific researcher rather than an engineer.

The Intelligence, Surveillance and Reconnaissance Division of DSTO for allowing me the opportunity to undertake these studies and the computing resources to undergo the work. The School of Electrical and Electronic Engineering at the University of Adelaide for their hospitality and learning environment.

To my wife, Lang for her love, support and understanding throughout my studies. For keeping me well nourished, encouraged when down and for listening to my presentations even when they made no sense to her, I thank you!

Abbreviations

CPU	Central Processing Unit
DSTO	Defence Science and Technology Organisation
EIF	Extended Information Filter
EKF	Extended Kalman Filter
EM	Expectation Maximisation
GA	Genetic Algorithm
GA-TSP	Genetic Algorithm for Travelling Salesmen Problem
ISRD	Intelligence, Surveillance and Reconnaissance Division
JPDA	Joint Probabilistic Data Association
KF	Kalman Filtering
LNN	Local Nearest Neighbour
NN-JPDA	Nearest Neighbour - Joint Probabilistic Data Association
pdf	probability density function
PF	Particle Filtering
PHD	Probability Hypothesis Density

pmf	probability mass function
PDA	Probabilistic Data Association
PMHT	Probabilistic Multi-Hypothesis Tracker
PMHT-c	Probabilistic Multi-Hypothesis Tracker with Classification measurements
PMHT-t	Probabilistic Multi-Hypothesis Tracker with Time measurements
PMHT-pp	Probabilistic Multi-Hypothesis Tracker with Path Planning
PMHT-pp-pf	Probabilistic Multi-Hypothesis Tracker Path Planner with Particle Filtering
POMDP	Partially Observable Markov Decision Processes
RMS	Root Mean Square
SLAM	Simultaneous Localisation and Mapping
TSP	Travelling Salesman Problem
UKF	Unscented Kalman Filter

Symbols

$\{\cdot\}^{\mathsf{T}}$	Matrix transpose operator.
C	The confusion matrix, gives the probability of observing a particular class given the true class.
c_{ij}	An element of the confusion matrix, $\mathsf{C} = \{c_{ij}\}$.
F	State transition matrix.
F_t	State transition matrix at scan t.
g_t	Measurement noise at time t.
H_t	Measurement matrix at scan t.
h_t	Nonlinear measurement matrix at scan t.
K	The set of all assignment indices over the data batch.
k_n	The assignment index for the nth measurement. Indicates which model is the true source for that measurement.
k_{nt}	The assignment index for the nth measurement at scan t. Indicates which model is the true source for that measurement.
k_{ntp}	The assignment index for the nth measurement at scan t from sensor p. Indicates which landmark is the true source for that measurement.
M	Total number of targets.

m	A target index, indicating target m.
N	Total number of measurements.
N_s	Total number of sample or particle points.
N_t	Total number of measurements at scan t.
N_{tp}	Total number of measurements from sensor p at scan t.
n	A measurement index, indicating measurement n
P	Total number of platforms.
P_d	The probability that a target is detected.
$P_{t\|t}$	The covariance of the state estimate at scan t.
$P_{t\|t-1}$	The covariance of the predicted state at scan t.
P_0	The covariance of the assumed distribution of the initial target state for the linear Gaussian case.
p	A platform index, indicating platform p in SLAM.
$Q\left(\cdot\|.^{(i)}\right)$	The EM auxiliary function that is maximised to obtain the iterative parameter estimates. It is a function of the true parameters and their estimates from the previous EM iteration.
Q	The process noise covariance
Q_C	The part of the auxiliary function dependent on the confusion matrix. This is maximised to find the confusion matrix estimate.
Q_t	The process noise covariance at scan t
Q_X	The part of the auxiliary function dependent on the target states, X
Q_{XY}	The part of the auxiliary function that couples the landmark and sensor states.
Q_Π	The part of the auxiliary function dependent on the assignment prior, Π.

Q_{Π}^{k}	The part of the auxiliary function dependent on the positional assignment prior, Π^{k}.
Q_{Π}^{τ}	The part of the auxiliary function dependent on the time stamp assignment prior, Π^{τ}.
Q_{τ}	The part of the auxiliary function dependent on the time stamp, τ.
q_x	The unit time variance of noise in x units.
q_y	The unit time variance of noise in y units.
R_t	The measurement covariance matrix at scan t.
$\hat{R}_t^m$	The synthetic sensor measurement covariance matrix for model m at scan t.
r	A measurement index for measurements at a particular scan.
S_t	The innovation covariance matrix at scan t. Represents the expected measurement scatter given the current state estimate and its covariance.
T	Total number of scans in the batch.
t	A time index, indicating scan number t.
u_t	Process noise at time t.
v	Probability of a correct measurement time being available.
v_t	Measurement innovation at scan t.
W_t	The Kalman Gain at scan t.
w_t	Measurement noise at time t.
w_t^i	The weight that a particular sample point i represents the state at time t.
w_{tm}	An assignment weight functions. The posterior probability function of a particular assignment to target m at scan t given the current estimated parameters.

w_{ntm}	An assignment weight. The posterior probability of a particular assignment between measurement n to target m at scan t given the current estimated parameters.
X	A set of all of the states of all models over the entire batch.
X^m	A set of all of the states of model m over the entire batch.
x_t	The state at scan t.
x_t^m	The state of model m at scan t.
x_t^i	Particle i to represent the state at scan t.
$\hat{x}_{t\|t-1}$	The predicted state at scan t.
$\hat{x}_t^m$	The state estimate for model m at scan t.
Y^p	A set of all of the states of platform p over all scans.
y_t^p	The state of platform p at scan t.
Z	A set of all of the measurements for the entire batch.
$Z^{(x)}$	A set of all of the positional measurements for the entire batch.
$Z^{(k)}$	A set of all of the classification measurements for the entire batch.
Z	A set of all of the measurements for the entire batch.
z_n	The nth measurement.
z_t	The measurement at scan t.
z_{ntp}^x	The nth positional measurement at scan t produced by sensor p.
z_{ntp}^k	The nth classification measurement at scan t produced by sensor p.
z_n	The nth measurement received by the sensor or the nth locale in the set of locales.

z_{nt}	The nth measurement at scan t.
z_n^x	The position of measurement n.
z_n^τ	The time stamp of measurement n.
$\tilde{z}_t^m$	The synthetic measurement for target m at scan t.
$\hat{z}_t$	The predicted measurement at scan t.
χ_t^i	Sample point i to represent the state at time t.
Δ	Pixel size of a uniform grid of locales.
Δt	Time difference between measurement updates.
η_n	Priority of locale n.
λ	Unknown parameter to estimate the inverse variance of the time stamp error.
μ	Unknown parameter to estimate the mean time stamp error.
$\phi_0^p(y_0^p)$	The prior probability density function for the state of sensor p.
$\phi_t^p(y_t^p \vert y_{t-1}^p)$	The evolution probability density function for sensor p at scan t.
Π	The set of all assignment priors for the batch.
Π^k	The set of all positional assignment priors for the batch.
Π^τ	The set of all time stamp assignment priors for the batch.
π_t^m	The assignment prior for model m at scan t.
π_{nm}^k	The assignment prior for measurement n to model m.
π_{nt}^τ	The assignment prior for measurement n at scan t.
$\psi_0(x_0)$	The prior probability density function for the state.
$\psi_0^m(x_0^m)$	The prior probability density function for the state of model m.

$\psi_t(x_t\|x_{t-1})$	The evolution probability density function for the target at scan t.
$\psi_t^m(x_t^m\|x_{t-1}^m)$	The evolution probability density function for model m at scan t.
ρ	Intensity of locales.
τ	The set of all time assignment indices over the data batch.
τ_n	The true collection time of measurement n.
θ^m	Associated class of each landmark measurement.
$\zeta\left(\boldsymbol{z}_t\|\boldsymbol{x}_t\right)$	The measurement probability density at scan t.
$\zeta\left(\boldsymbol{z}_{nt}\|\boldsymbol{x}_t^m\right)$	The measurement probability density for model m at scan t.

Publications

1. B Cheung, S J Davey, and D A Gray, *Combining PMHT with classifications to perform SLAM*, Proceedings of the 12th International Conference on Information Fusion (Seattle, US), July 2009.

2. B Cheung, S J Davey, and D A Gray, *PMHT for Multiple Platform Path Exploration Planning*, Proceedings of the 5th International Conference on Intelligent Sensors, Sensor Networks and Information Processing (Melbourne, Australia), December 2009.

3. B Cheung, S J Davey, and D A Gray, *PMHT for Tracking with Timing Uncertainty*, Proceedings of the 13th International Conference on Information Fusion (Edinburgh, UK), July 2010.

4. B Cheung, S J Davey, and D A Gray, *Comparison of the PMHT Path Planning Algorithm with the Genetic Algorithm for Multiple Platforms*, Proceedings of the 13th International Conference on Information Fusion (Edinburgh, UK), July 2010.

5. B Cheung, S J Davey, and D A Gray, *PMHT Particle Filter for Indoor Multiple Platform Path Exploration Planning*, Proceedings of the 7th Defence Applications of Signal Processing (Coolum, Australia), July 2011.

6. B Cheung, S J Davey, and D A Gray, *PMHT Path Planning in a non-Homogeneous Environment*, Proceedings of the 7th International Conference on Intelligent Sensors, Sensor Networks and Information Processing, December 2011.

7. B Cheung, S J Davey, and D A Gray, *PMHT for Multiple Platform Path Planning*, Journal paper, DSTO peer reviewed and to be submitted to IEEE Robotics, 2011.

8. B Cheung, S J Davey, and D A Gray, *PMHT with Time Uncertainty*, Journal paper, DSTO peer reviewed and to be submitted to IEEE Transactions on Aerospace and Electronic Systems, 2011.

Chapter 1

Introduction

Tracking is the task of recursively estimating a target's state from a set of noisy and ambiguous measurements. A typical tracking filter requires the motion model of the target and a model of the measurement process. The state of the target is then estimated by applying these models to the sensor data. This thesis is primarily concerned with a particular tracking algorithm called the Probabilistic Multi-Hypothesis Tracker (PMHT). The PMHT is a multi-target tracker introduced by Streit and Luginbuhl [SL95].

Modern surveillance systems use a variety of sensors depending on the situation and environment. Examples of sensors include radars for long range surveillance, electro-optics for video surveillance, infrared for night surveillance and sonar for underwater surveillance. Although each sensor has its own capabilities and measurement characteristics, the same tracking techniques can be used. The tracking filter uses the supplied measurement data and previous knowledge (such as the target models and sensor characteristics) to detect, locate and track the targets.

It is customary to assume that measurement data is collected at known times. As the trend in surveillance systems heads towards the usage of low cost sensors and distributed sensing, the timing information may not be so reliable leading to uncertainty in the measurement collection times. The first focus of this thesis is to account for this time uncertainty by introducing a novel extension to the PMHT algorithm.

The second focus of this thesis is the application of tracking techniques in the field of robotics. The task of navigating platforms through an area has many applications including search and rescue, coordinated surveillance, exploration and resource dissemination. There

are many strategies to control a single moving platform to intelligently explore an environment. These strategies range from the basic, such as constraining the platform to only move towards areas yet to be explored [Yam97], to more complex strategies such as decreasing the localisation error of the platform while increasing the potential information gain of the sensors [FLS99]. Most of these strategies only look one step ahead and may not explore the whole area optimally. This problem is even more difficult when multiple platforms are considered. This thesis introduces a new approach to plan paths for multiple platforms to explore an unknown area cooperatively, by treating the navigation problem as a multi-target tracking system.

An ideal autonomous vehicle has the ability to simultaneously:

- Gain information about its environment;
- Navigate through the environment based on the information gained;
- Work without human intervention.

Much research has been focused to achieve these goals. The task of building a map in an unknown environment and using that map to navigate is a central problem in mobile robotics research. This problem is known as Simultaneous Localisation and Mapping (SLAM) [Thr02].

In an ideal environment, the robot would have access to measurements of its absolute position, such as via a GPS receiver, but there are many situations where this is not feasible, such as indoors or where the GPS accuracy is not sufficient. The GPS signal can also be easily jammed. Some extreme situations where GPS may not be available include exploration of underground mines, underwater exploration in unstructured environments or even the use of autonomous rovers exploring the surface of planet Mars.

Autonomous vehicles use a range of environmental sensors to gain information about their environment. These sensors commonly include cameras, laser range finders and sonar. These sensors can provide information such as target detection or provide positional information of the targets and of the platform itself. In many cases additional information is available that could be used to classify the measurements. Classification may be based on colour, size or shape of the objects to be mapped and the use of classification data can assist recognition of the target. This thesis considers using classification information with tracking to perform SLAM more effectively.

1.1 Contributions

The most significant contributions made by this thesis are as follows.

Chapter 3:

- Extension of the PMHT to track using measurements with noisy time information, referred to as the PMHT-t.
- Extension of the PMHT-t to simultaneously estimate the timing error parameters, to associate targets with measurements and to associate measurements with time.

Chapter 4:

- Novel use of the PMHT-t extension to perform path planning for multiple platform exploration, referred to as the PMHT-pp.
- Use of the Genetic Algorithm to solve the multiple Travelling Salesmen Problem (TSP) combined with the PMHT-pp to smooth the trajectories of the platform solutions generated by the Genetic Algorithm.
- Use of the PMHT-pp with a particle filter state estimator for indoor path planning with obstructions.
- Extension of the PMHT-pp to plan paths driven by a non-homogeneous priority intensity map.

Chapter 5:

- Use of classification information to improve data association in a SLAM context using PMHT with classification, referred to as the PMHT-c.

1.2 Thesis Overview

The structure of the thesis is now described. Chapter 2 presents background information about multi-target tracking. The multi-target tracking problem has two major subproblems, data association and state estimation. A description of various methods to perform data association is given. Several simple models of target motion are provided, which are then used in the last section which covers different state estimation methods.

A standard tracking filter performs recursive target state estimation given a set of noisy position measurements collected at known times. It is customary to assume that there is no error in the time information, but under some situations, such as distributed sensing, the timing information may not be completely reliable. Chapter 3 derives an extension to the PMHT that introduces an assignment index that identifies the true time at which a measurement was collected, referred to as the PMHT-t. Simulations are used to demonstrate the effectiveness of the PMHT-t to track using measurements with stochastic time delay errors. A further simulation compares the PMHT-t algorithm with several other algorithms to deal with a scenario where true measurement times are available occasionally while noisy measurement times are available for all measurements. The PMHT-t is extended to simultaneously estimate the timing error parameters and track the target.

Chapter 4 uses a method based on the PMHT-t as a new approach to multiple platform path planning. Given known initial conditions for a set of platforms, (such as position and speed), and a set of discrete locales of interest, the PMHT path planner, referred to as the PMHT-pp, is used to design trajectories for the platforms to cooperatively visit the locales.

The novelty in the PMHT-pp approach is to treat the locales as measurements and the platforms as targets and then employ a multi-target tracking algorithm to perform data association, namely to associate the locales with platforms. The trajectories of the platforms are constrained by their dynamic models, resulting in feasible planned paths. The locales have no intrinsic temporal relationship, so there is no known sequence of locales to be visited by a platform. The PMHT-pp uses a hidden variable to hypothesise time-stamps to associate locales to platforms and times simultaneously.

Chapter 4 derives the PMHT-pp algorithm and discusses the tradeoffs in the system and measurement equation parameters to constrain the trajectories of the paths. The PMHT-pp algorithm is also compared to a Genetic Algorithm based solution to solve the TSP. This comparison includes a technique to use the PMHT-pp to create feasible trajectories for platforms given an ordered list of locales to visit. The chapter then discusses a sliding batch version of the PMHT-pp algorithm, which allows for the revisiting of areas to reduce platform localisation errors.

The use of the PMHT-pp algorithm with a particle filter for state estimation is then used to explore an unknown area with obstructions, such as an indoor environment. The chapter concludes with an extension to the PMHT-pp to create trajectories using a non-homogeneous intensity map.

Chapter 5 uses classification information received with measurements to improve the performance of SLAM. The problem referred to as SLAM requires estimation of unknown target locations when the platform location knowledge is unreliable. It is a technique often associated with autonomous platforms that carry a variety of complementary sensors. Beside target detection and platform positional information, these sensors, such as laser ranging and cameras, can often provide perceived classification information that is generally not utilised by the data association algorithm. Chapter 5 uses the classification extension to PMHT, referred to as PMHT-c to use this classification information to aid the data association. The PMHT-c SLAM algorithm is implemented on both simulated and real data SLAM scenarios to illustrate the performance improvement.

Chapter 6 presents a summary of the thesis and potential directions for future work.

Chapter 2

Background

The aim of a multi-target tracking algorithm is to estimate the states of the targets from a set of noisy and ambiguous measurements. It is usually assumed that the sensor provides an unlabelled set of measurements. These measurements reflect the position of the target combined with noise that may be environmental or caused by the sensor itself. Typically the sensor also reports false alarms that are measurements not due to any target.

The multi-target tracking problem has two major sub-problems which must be solved simultaneously [BSF88],[BP99]. The first sub-problem is to determine which measurement was caused by each of the targets. This is often called "data association" and there are many techniques to solve it.

The second sub-problem is to estimate the state of the target using the set of associated measurements. In many applications, estimates are required in real-time and an iterative estimator can be used to estimate the current state based on the associated measurements and the previous target state. In other cases, data may be accumulated over a series of time points and a batch estimator, or a smoother, can be used to estimate the target state using the set of associated measurements.

2.1 Data Association

Data association is the process by which an algorithm assigns measurements to targets. The assignment of measurements to targets can be done in a hard or soft way. Hard assignment is when a decision is made about whether a measurement was produced by a particular target or

not and the decision can only be yes or no. The target state estimates are then updated assuming that the assigned measurements are the correct measurement for the targets.

A soft assignment is where multiple measurements may be assigned to each target with a certain probability. Under a linear measurement model, the set of assigned measurements can then be combined using the assignment probabilities to form a pseudo measurement to update the target state estimate.

Data association can be applied as a single-target or multi-target approach. In single-target data association, it is assumed that the assignment of measurements is independent for different tracks. In contrast, multi-target data association algorithms jointly assign measurements to multiple tracks simultaneously. To reduce computational complexity, methods such as limiting the area of influence of a track can be used.

Data association is described in detail in many tracking texts, for example [BSF88], [BP99] and [BSL88]. This section will discuss the different types of data association that will be used in this thesis.

2.1.1 Nearest Neighbour

The nearest neighbour algorithm is the simplest single-target hard assignment data association algorithm [BSL88]. The nearest neighbour approach forms a predicted measurement for each track and compares this with the actual sensor measurements. The sensor measurement closest to the predicted measurement is associated with the track.

A simple extension of the nearest neighbour association algorithm is the Local Nearest Neighbour where the closest measurement is only associated if it is within a gate distance from the target [BP99]. This extension avoids the situation where a target track is associated with an infeasibly distant measurement due to the target not being detected by the sensor. Without the gate test, the filter update may cause the track to diverge from the target trajectory if an outlier is associated.

2.1.2 Probabilistic Data Association Filter

The Probabilistic Data Association Filter (PDA or PDAF) is a popular association algorithm due to its simplicity and speed [BST75]. PDA differs from nearest neighbour in that it allows for uncertainty in the decision about which measurement belongs to a target. PDA assumes that

the target state density can be approximated by its first two moments, and produces an updated Gaussian pdf approximation. This approach has been demonstrated to improve tracking performance over nearest neighbour association. However, if the pdf of the target state is strongly multi-modal, then the PDA is clearly throwing away information and the performance can be adversely affected.

The PDA is a single-target algorithm which performs soft assignment. The PDA has been extended to handle multiple targets and this extension is referred to as the Joint-PDA (JPDA) [BSL88]. The JPDA explicitly enumerates the joint association events of all of the targets, the number of which grows combinatorially with the number of targets and measurements. Clustering is usually required to achieve a physically practical algorithm.

To reduce the computational complexity there is nearest neighbour JPDA (NN-JPDA) [Fit90], also known as "cheap JPDA". Cheap JPDA approximates JPDA by using an approximate method to find the single best joint association event. The approximate method for finding the best joint association event is based on approximating the joint association probability by a product of marginal probabilities and finding the maximum using iterative marginal maximisation.

2.1.3 Probabilistic Multi-Hypothesis Tracker

The Probabilistic Multi-Hypothesis Tracker (PMHT) is a method that performs soft assignment and is a multi-target data association technique. In standard Bayesian approaches, the assignment of measurements to targets is constrained so that each target is allowed to form at most one measurement. PMHT is a very different approach in this respect. In the PMHT framework, the true source of each measurement is treated as an independent random process with an associated probability mass function. This allows PMHT to treat the association of measurements and targets as independent for different measurements. The marginal assignment probability is used for data association which leads to the soft assignment. The problem of association and estimation becomes a joint estimation process of two sets of random variables: continuous target states and discrete measurement assignments.

The PMHT, developed by Streit and Luginbuhl [SL95], is a data association algorithm derived from the application of the Expectation Maximisation (EM) algorithm [DLR77] to target tracking. The PMHT uses EM to model the assignment of measurements to targets as hidden variables and estimates target states by taking the expectation over the assignments.

The major advantage of using PMHT is that the computational complexity is linear in the

number of targets, measurements and time steps, unlike other association algorithms which require enumeration of joint association events, such as the JPDA above. This allows the algorithm to be implemented without approximation and allows for efficient smoothing over time batches when the application requires.

The standard PMHT generally performs data association with kinematic measurements (such as position, speed and acceleration) of the target state. In practice, additional information may be available to the tracker which can improve the measurement association. This information can take the form of non-kinematic information which can be received from alternative sensors. The non-kinematic information can be treated as a classification measurement and has been used in the PMHT framework to improve data association. This extension of the algorithm is referred to as PMHT-c [DGS02].

2.2 State Estimation

The second part of the multi-target tracking problem is state estimation. It is assumed that the target can be described by a list of parameters of interest and these are combined to form a state vector, which evolves over time and is often only partially observed by the sensor. The probability density of the state is of interest because it can be used to extract MAP or ML estimates. State estimation is therefore the problem of calculating the probability density of the state conditioned on the sensor data.

The tracking filter performs recursive target state estimation using: (i) a target motion model; (ii) a sensor measurement model and (iii) target originated measurements. The choice of tracking filter is dependent on the choice of the coordinate system and the linearity of the problem.

In the case where the system is linear and the random elements are Gaussian random variables, an optimal closed form estimator exists and is the Kalman Smoother.

For non-linear problems, one method is to approximate the non-linear functions by a truncated Taylor series, resulting in the Extended Kalman Filter [Jaz70]. Alternatively, the pdf can be approximated as a Gaussian and the mean and covariance can be estimated numerically using a set of sample points, a method referred to as the Unscented Kalman Filter [JU04]. However, for problems where the pdfs to be approximated are highly non-Gaussian, it may not be desirable to make analytic approximations. In this case, Monte Carlo integration techniques can be used to represent the state pdf by a collection of random samples. This method is referred to as

the particle filter [DFG01] and is capable of describing an arbitrary process with non-Gaussian noise with accuracy linked to the number of samples used.

This section will define the state estimation problem and then present different algorithms to solve both linear and non-linear problems.

2.2.1 Problem Definition

Assume that there is a sensor that is tracking a target by collecting measurements produced by the target at discrete scans. Let the total number of scans be T.

Let the state of the single target at time t be denoted x_t, and let X denote the set of all states for the target, i.e. $X = \{x_t\}$ for $t = 1 \dots T$.

It is assumed that the prior distribution of the target state is known and is given by $\psi_0(x_0)$. The target dynamics are also assumed to be known and can be described by the evolution probability density function (pdf) $\psi_t(x_t|x_{t-1})$.

Let the measurement at time t for the target received by the sensor be denoted by z_t. Unless otherwise stated, the state estimation filters in this chapter will assume that the data association between the measurement and track is known.

Let Z denote the set of all measurements. The sensor observation process is described by a known measurement pdf that is denoted by $\zeta(z_t|x_t)$. The particular form of the measurement pdf may be dependent on the sensors used in the application.

2.2.2 Target Dynamic Models

To track a target, the target is represented by a state vector. This state vector may include the targets position, velocity and other information. An example of a state vector is

$$x_t = [x, \dot{x}, y, \dot{y}]^{\mathsf{T}} \tag{2.1}$$

where x and y are the positions in the x and y axis and $\dot{x}$ and $\dot{y}$ are the respective velocities.

A target dynamic model is used to describe the evolution of the target state with respect to time. Surveys of the many different dynamic models that can be used in tracking are presented in [BP99] and [LJ00]. However, the most commonly used model is the constant velocity Gaussian

model. For this and the above state vector, the process equation can be written as

$$x_{t+1} = Fx_t + u_t, \tag{2.2}$$

where

$$F = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.3}$$

and u_t is a zero mean Gaussian noise process with covariance

$$Q = \begin{bmatrix} q_x \frac{1}{3}\Delta t^3 & q_x \frac{1}{2}\Delta t^2 & 0 & 0 \\ q_x \frac{1}{2}\Delta t^2 & q_x \Delta t & 0 & 0 \\ 0 & 0 & q_y \frac{1}{3}\Delta t^3 & q_y \frac{1}{2}\Delta t^2 \\ 0 & 0 & q_y \frac{1}{2}\Delta t^2 & q_y \Delta t \end{bmatrix}. \tag{2.4}$$

Δt is defined as the sample period, q_x and q_y are the unit time variance of acceleration noise in the x and y units respectively. This model assumes that the target moves independently in x and y, which results in the block-diagonal form of F and Q.

Another commonly used model is the constant acceleration model. The state vector now includes the position, velocity and acceleration

$$x_t = [x, \dot{x}, \ddot{x}, y, \dot{y}, \ddot{y}]^{\mathsf{T}}, \tag{2.5}$$

where $\ddot{x}$ is the acceleration in the x axis.

Like the constant-velocity model, the constant-acceleration model is linear and the process equation is the same as 2.2, only with different state transition and measurement covariance matrices, given by

$$F = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ 0 & 1 & \Delta t & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.6}$$

$$Q = \begin{bmatrix} q_x\frac{1}{20}\Delta t^5 & q_x\frac{1}{8}\Delta t^4 & q_x\frac{1}{6}\Delta t^3 & 0 & 0 & 0 \\ q_x\frac{1}{8}\Delta t^4 & q_x\frac{1}{3}\Delta t^3 & q_x\frac{1}{2}\Delta t^2 & 0 & 0 & 0 \\ q_x\frac{1}{6}\Delta t^3 & q_x\frac{1}{2}\Delta t^2 & q_x\Delta t & 0 & 0 & 0 \\ 0 & 0 & 0 & q_y\frac{1}{20}\Delta t^5 & q_y\frac{1}{8}\Delta t^4 & q_y\frac{1}{6}\Delta t^3 \\ 0 & 0 & 0 & q_y\frac{1}{8}\Delta t^4 & q_y\frac{1}{3}\Delta t^3 & q_y\frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & q_y\frac{1}{6}\Delta t^3 & q_y\frac{1}{2}\Delta t^2 & q_y\Delta t \end{bmatrix}. \tag{2.7}$$

It is observed that the covariance models (2.4) and (2.7) follow from a continuous time target state evolution that is discretely sampled [BSF88].

A third model that is often used for moving targets is the constant speed and constant turn rate model. The state vector consists of the position, x and y, heading, θ, the speed, s and the steering angle, $\dot{\theta}$, namely

$$x_t = \left[x, y, \theta, s, \dot{\theta}\right]^{\mathsf{T}} \tag{2.8}$$

The heading is the time integral of the steering angle and the position is the time integral of the velocity, which is a function of both the speed and instantaneous heading. The resulting non-linear state transition vector function is

$$F(x_t) = \begin{bmatrix} x + \Delta t s\cos\theta - \frac{\Delta t^2}{2}s\dot{\theta}\sin\theta \\ y + \Delta t s\sin\theta + \frac{\Delta t^2}{2}s\dot{\theta}\cos\theta \\ \theta + \Delta t\dot{\theta} \\ s \\ \dot{\theta} \end{bmatrix}. \tag{2.9}$$

Note that the state transition matrix is time varying for this model even though the sampling rate is constant.

The measurement covariance matrix is given by

$$Q(x_t) = A_t(x_t)B_t{B_t}^{\mathsf{T}}{A_t(x_t)}^{\mathsf{T}} \tag{2.10}$$

where

$$A_t(x_t) = \begin{bmatrix} \sigma_s \cos\theta & 0 & 0 & 0 \\ \sigma_s \sin\theta & 0 & 0 & 0 \\ 0 & \sigma_{\dot{\theta}} & 0 & 0 \\ 0 & 0 & \sigma_s & 0 \\ 0 & 0 & 0 & \sigma_{\dot{\theta}} \end{bmatrix}, \tag{2.11}$$

and

$$B_t = \begin{bmatrix} \sqrt{\frac{\Delta t^3}{3}} & 0 & 0 & 0 \\ \frac{\sqrt{3\Delta t}}{2} & \frac{\sqrt{\Delta t}}{2} & 0 & 0 \\ 0 & 0 & \sqrt{\frac{\Delta t^3}{3}} & 0 \\ 0 & 0 & \frac{\sqrt{3\Delta t}}{2} & \frac{\sqrt{\Delta t}}{2} \end{bmatrix}, \tag{2.12}$$

where σ_s and $\sigma_{\dot{\theta}}$ is the standard deviation in speed and steering angle respectively.

2.2.3 Measurement Model

Recall that there is a set of measurements produced by a single target with no measurement source ambiguity being tracked. These measurements are related to the target state vector via the measurement equation

$$z_t = h_t(x_t, w_t), \tag{2.13}$$

where the source of measurement z_t at time t is the target and w_t is random noise. The measurement function h_t is assumed to be known. In many tracking problems, the measurement function h_t is non-linear as it translates between different coordinate systems. For example, a sensor may receive observations in a polar coordinate system (such as range and bearing) and a target state position described in Cartesian coordinates (such as north and east).

However, the measurement process in tracking problems is often assumed to have a linear measurement model. For a linear Gaussian measurement model, equation (2.13) becomes

$$z_t = H_t x_t + g_t, \tag{2.14}$$

where H_t is an appropriately sized matrix and g_t is zero mean Gaussian noise with covariance R_t.

In many applications, measurements contain only observations of the target position. This leads to the measurement function H_t being a linear translation of the target state. For example,

the measurement function for the constant velocity target model from Section 2.2.2 is

$$H_t = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. \tag{2.15}$$

2.2.4 Kalman Filter

If the system and measurement processes are linear and the random elements are Gaussian, then the posterior state pdf is also Gaussian and can be fully described by its mean and variance. The Kalman filter is an optimal recursive estimator for the mean and variance of the posterior state pdf. The Kalman filter assumes that the initial mean and variance of the posterior state pdf are known.

The Kalman filter consists of two steps, a prediction and an update step. The first step calculates the predicted mean and variance based on the previous data. The update step, also known as the correction step, adjusts the predicted mean and variance using the current measurement.

The prediction step predicts the new mean and variance by projecting ahead using the transition matrix, given by the following equations:

$$\hat{x}_{t|t-1} = F_t \hat{x}_{t-1|t-1} \tag{2.16}$$

$$P_{t|t-1} = F_t P_{t-1|t-1} F_t^{\mathsf{T}} + Q_t \tag{2.17}$$

where $\hat{x}_{t|t-1}$ is the predicted mean and $P_{t|t-1}$ the predicted variance.

The prediction step also estimates the mean and covariance of the measurement z_t at scan t, conditioned on the predicted state and its associated error covariance. The mean of the conditioned measurement pdf is given by

$$\hat{z}_t = H_t \hat{x}_{t|t-1} \tag{2.18}$$

and its covariance is given by

$$S_t = H_t P_{t|t-1} H_t^{\mathsf{T}} + R_t \tag{2.19}$$

where R_t is the measurement noise covariance.

The update step involves correcting the predicted state using the measurement of the target. The update step calculates the measurement innovation, given by

$$\nu_t = z_t - \hat{z}_t, \tag{2.20}$$

which is the difference between the expected measurement and the actual observed measurement.

The Kalman gain W_t is calculated using

$$W_t = P_{t|t-1} {H_t}^{\mathsf{T}} {S_t}^{-1}. \tag{2.21}$$

The correction step then concludes by forming the new state mean and covariance:

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + W_t \nu_t, \tag{2.22}$$

$$P_{t|t} = P_{t|t-1} - W_t H_t P_{t|t-1}. \tag{2.23}$$

2.2.5 Extended Kalman Filter

The extended Kalman filter (EKF) is a common approach to implement a recursive non-linear estimation filter [Jaz70]. This approach is applicable to non-linear models with additive Gaussian noise. It is a linearisation technique based on a first order truncation of the Taylor series expansion of the non-linear system and measurement functions about the current estimate of the state. The system and measurement functions f_t and h_t need not be linear functions, but it is assumed they are differentiable functions. These functions f and h cannot be directly applied to the covariance, but instead a matrix of Jacobian partial derivatives is computed.

The prediction equations for the state and covariance are similar to the standard Kalman filter,

$$\hat{x}_{t|t-1} = f_{t-1}(\hat{x}_{t-1|t-1}) \tag{2.24}$$

$$P_{t|t-1} = F_{t-1} P_{t-1|t-1} {F_{t-1}}^{\mathsf{T}} + Q_t, \tag{2.25}$$

with the state transition matrix given by

$$F_{t-1} = \left. \frac{\partial f_{t-1}(x_{t-1})}{\partial x_{t-1}} \right|_{x_{t-1}=\hat{x}_{t-1|t-1}} \tag{2.26}$$

Similarly for the measurement equation, a Taylor series expansion of the measurement function is taken,

$$\hat{z}_t = h_t(x_t) \tag{2.27}$$

with covariance and Jacobian observation function

$$S_t = H_t P_{t|t-1} H_t^{\mathsf{T}} + R_t \tag{2.28}$$

$$H_t = \left.\frac{\partial h_t(x_t)}{\partial x_t}\right|_{x_t=\hat{x}_{t|t-1}} \tag{2.29}$$

The measurement innovation calculated by the update step is given by

$$\nu_t = z_t - \hat{z}_t \tag{2.30}$$

and the Kalman gain W_t is calculated using

$$W_t = P_{t|t-1} H_t^{\mathsf{T}} {S_t}^{-1}. \tag{2.31}$$

The target state and covariance is then updated using the approximations, similar to the standard Kalman filter,

$$\hat{x}_{t|t} = \hat{x}_{t|t-1} + W_t \nu_t, \tag{2.32}$$

$$P_{t|t} = P_{t|t-1} - W_t H_t P_{t|t-1} \tag{2.33}$$

The prediction and update steps are very similar to the standard Kalman filter algorithm and so are very popular in solving non-linear estimation problems. The most important factor in applying the EKF is the linear approximation to the non-linear functions. If the linearisation is poor, bad filter performance such as significant biases or filter divergence is expected. It is also possible that approximating the posterior distribution by a Gaussian distribution may be a significant distortion of the true underlying pdf.

For best performance from the EKF, small nonlinearities, relatively small system noise and a good initial state estimates are required.

2.2.6 Unscented Kalman Filter

Another popular non-linear version of the Kalman Filter is the Unscented Kalman Filter (UKF). Unlike the EKF, the UKF does not analytically approximate the non-linear system and measurement functions. Instead, it uses an unscented transform, developed in [JU04], to approximate the posterior pdf by a Gaussian density which is represented by a set of deterministically

chosen sample points. The set of N_s sample points, $\chi^i, i = 1, \ldots, N_s$, and their weights, $w^i, i = 1, \ldots, N_s$, capture the true mean and covariance of the state pdf.

The predicted density $p(x_t|Z_{1:t-1}) \approx N(x_t; \hat{x}_{t|t-1}, P_{t|t-1})$ is represented by propagating the set of N_s sample points via the transition function:

$$\chi^i_{t|t-1} = f_{t-1}(\chi^i_{t-1}). \tag{2.34}$$

Similar to the other Kalman filter based approaches, the UKF has a prediction and an update step. In the prediction step, the target state and covariance are calculated using the weighted sum of the sample points from 2.34, which gives:

$$\hat{x}_{t|t-1} = \sum_{i=0}^{N-1} w^i_{t-1}(\chi^i_{t|t-1}) \tag{2.35}$$

$$P_{t|t-1} = Q_{t-1} + \sum_{i=0}^{N-1} w^i_{t-1}[\chi^i_{t|t-1} - \hat{x}_{t|t-1}][\chi^i_{t|t-1} - \hat{x}_{t|t-1}]^\mathsf{T} \tag{2.36}$$

The predicted measurement is calculated using the weighted set of predicted sample points:

$$\hat{z}_t = \sum_{i=0}^{N-1} w^i_{t-1} h_t(\chi^i_{t|t-1}) \tag{2.37}$$

The update step takes similar form to the other Kalman filtering update step:

$$\hat{x}_t = \hat{x}_{t|t-1} + W_t \nu_t, \tag{2.38}$$

$$P_t = P_{t|t-1} - W_t H_t P_{t|t-1}. \tag{2.39}$$

where the differences are

$$\nu_t = z_t - \hat{z}_t \tag{2.40}$$

$$W_t = P_{xz} S_t^{-1} \tag{2.41}$$

$$S_t = R_t + P_{zz} \tag{2.42}$$

$$P_{xz} = \sum_{i=0}^{N-1} w_{t-1}^i (\chi_{t|t-1}^i - \hat{x}_{t|t-1})(h_t(\chi_{t|t-1}^i) - \hat{z}_t)^\mathsf{T} \tag{2.43}$$

$$P_{zz} = \sum_{i=0}^{N-1} w_{t-1}^i (h_t(\chi_{t|t-1}^i) - \hat{z}_t)(h_t(\chi_{t|t-1}^i) - \hat{z}_t)^\mathsf{T}. \tag{2.44}$$

At the end of the update step, the target state posterior density and covariance are reconstructed from the sample points and is assumed to be Gaussian. One advantage of the UKF over the EKF is not having to calculate the Jacobians to linearise the non-linear functions. Thus a wider variety of processes are admissible. Also, a more accurate estimate of the exact mean and covariance can be obtained by simply increasing the number of sample points. Methods for improving the accuracy of the EKF are more complicated and usually ad hoc.

2.2.7 Particle Filter

Particle filtering is a popular scheme for tracking targets in non-linear problems [DFG01]. Unlike Kalman-based filters which summarise the target density using a mean and variance, the particle filter describes the density directly using a set of randomly chosen support points and associated weights. The set of particles is then propagated over time using the target motion models. The main advantage of the particle filter is the ability to track non-linear and non-Gaussian targets.

The particle filter uses a set of particles $\{x_t^i, w_t^i\}_{i=1}^{N_s}$ to characterise the posterior pdf $p(x_t|z_t)$, where $\{x_t^i\}, i = 1, ..., N_s$ is a set of N_s support points with associated weights $\{w_t^i\}, i = 1, ..., N_s$. The weights are normalised such that $\sum_i w_t^i = 1$. Therefore a discrete weighted approximation to the true posterior density, $p(x_t|z_t)$ is given by

$$p(x_t|z_t) \approx \sum_{i=1}^{N_s} w_t^i \delta(x_t - x_t^i). \tag{2.45}$$

The weights are chosen using the principle of importance sampling [AMGC02]. This in-

volves choosing an importance density $q(x_t)$ that can be used to easily generate and propose samples from, which is difficult to do from density $p(x_t)$.

For the case where only a filtered estimate of $p(x_t|z_{1:t})$ is required at each time step, [AMGC02] provides an approximation to the posterior filtered density. In summary, the weights for the posterior filtered density $p(x_t|z_{1:t})$ approximation using equation (2.45) are defined as

$$w_t^i \propto w_{t-1}^i \frac{p(z_t|x_t^i)p(x_t^i|x_{t-1}^i)}{q(x_t^i|x_{t-1}^i, z_t)} \tag{2.46}$$

It can be shown that as the number of particles increases $N_s \to \infty$, the approximation (2.45) approaches the true posterior density $p(x_t|z_t)$ [AMGC02]. Particle filters are currently limited to a smaller dimensional state space, which may be overcome with a higher number of particles.

The particle filter algorithm follows a similar prediction and update step as the other state estimation filters. The filter starts with a set of particle points and weights, $\{x_{t-1}^i, w_{t-1}^i | i = 1, \ldots, N_s\}$, which describes the posterior density at $t-1$ as

$$p(x_{t-1}|z_{t-1}) \approx \sum_{i=1}^{N_s} w_{t-1}^i \delta(x_{t-1} - x_{t-1}^i). \tag{2.47}$$

The particles at time $t-1$ are propagated using the mean proposal distribution, q to form a new set of particles at time t, which is the prediction step. The weights at time t are a function of the process dynamics and the measurement likelihood 2.46. This new set of particles and weights forms $\{x_t^i, w_t^i | i = 1, \ldots, N_s\}$ which gives an approximation for the posterior density at time t, which is the update step.

One major problem with particle filtering is known as the degeneracy problem. Degeneracy describes the situation where one particle dominates and the weights of the others are negligible. This can be shown to be guaranteed to occur after a finite, and often small, number of filter recursions [GB01]. With this situation, the particles do not provide sufficient samples to estimate the target pdf. To get around this problem, resampling is used. Resampling eliminates samples with low weights and multiplies samples with high weights. It involves mapping the set of particles and weights into a new set of random particles with uniform weights, such that the estimated target pdf is the same. More information about resampling can be found in [AMGC02].

2.3 Probabilistic Multi-Hypothesis Tracker

The previous state estimation algorithms assumed that the track to measurement assignments were known. The PMHT models these unknown assignments directly and performs both data association and multi-target state estimation simultaneously. The problem of data association and multi-target state estimation is now defined here as an extension to the problem definition in Section 2.2.1.

2.3.1 Problem Definition for multi-target state estimation, data association and clutter modelling

Assume that there is a sensor that is tracking M targets. At discrete scans, it collects measurements, some of which are due to the targets, and some of which are false alarms, referred to as clutter. Let the total number of measurements be N and the number of scans be T.

Let the state of target m at time t be denoted x_t^m, and let X^m denote the set of all states for target m, i.e. $X^m = \{x_t^m\}$ for $t = 1 \dots T$. Similarly, let the set of all states across the targets be denoted by $X = \{X^m\}$ for $m = 1 \dots M$.

It is assumed that the prior distribution of the state of each target is known and is given by $\psi_0^m(x_0^m)$ for target m. The target dynamics are also assumed to be known and can be described by the evolution probability density function (pdf) $\psi_t^m(x_t^m|x_{t-1}^m)$.

Let the nth measurement received at time t by the sensor be denoted by z_{nt}. At time t there are N_t measurements received by the sensor. This number of measurements N_t may be zero.

Let Z denote the set of all measurements. The sensor observation process is described by a known measurement pdf for the nth measurement at time t and is denoted by $\zeta(z_{nt}|x_t^m)$. The particular form of the measurement pdf may be dependent on the sensors used in the application.

Within the set of N measurements at time t, let the true source of measurement n be $k_{nt} \in 0 \dots M$, where $k_{nt} = 0$ means that the measurement is a false alarm and $k_{nt} = m$ implies that measurement n is an observation of target m. This assignment k_{nt} is unknown and is treated as a random variable with prior π_t^m. The prior distribution is also unknown. Let K denote the set of all measurement-to-target assignments and Π denote the collection of assignment priors.

2.3.2 PMHT for multi-target tracking

The PMHT algorithm is a method for finding the best estimate of the target states, X, when the measurement source K is unknown. It does this by treating the assignments, K as missing data using EM. In EM terminology, the complete data are (X, K, Z), the incomplete data are (X, Z), and K is the missing data. The auxiliary function is the expectation of the complete data log-likelihood over the missing data, which takes the form:

$$Q(X, \Pi|\hat{X}(i), \hat{\Pi}(i)) = \sum_K P(K|\hat{X}(i), Z; \hat{\Pi}(i)) \log P(X, K, Z; \Pi), \tag{2.48}$$

where the summation is over all permutations of the assignment variable K. The PMHT is an iterative algorithm, which asymptotically approaches a local maximum of the EM auxilliary function by refining estimates for the states X and the parameters Π. At the ith iteration, the estimated states and parameters are denoted as $\hat{X}(i)$ and $\hat{\Pi}(i)$ respectively. The iterations are repeated until the auxiliary function converges and the algorithm's output is the final state estimate.

This auxiliary function $Q(.)$ can be maximised using any appropriate estimator. It can be shown that the auxiliary function is equivalent to the log-likelihood of a known assignment problem with synthetic measurements determined by the expectation step [SL95]. Thus for linear Gaussian cases, the Kalman filter may be used to solve the equivalent problem. For non-linear problems such as tracking with range and bearing measurements, a non-linear filter must be used, such as an Extended Kalman Filter, Unscented Kalman Filter or a particle filter. Some non-linear problems can be turned into continuous mixture problems and solved by EM .

The following independence assumptions are made:

- all state sequences are independent of each other;
- the (unknown) true assignments are independent, identically distributed random variables with a prior probability mass $\pi_t^m \equiv P(k_{nt} = m)$;
- all measurements are conditionally independent given the assignments and the states of the targets.

Due to the independence assumptions, the complete data likelihood becomes:

$$P(X, K, Z; \Pi) = P(X)P(K; \Pi)P(Z|X, K), \tag{2.49}$$

where

$$P(X) = \prod_{m=1}^{M} \psi_0^m(x_0^m) \prod_{t=1}^{T} \psi_t^m(x_t^m|x_{t-1}^m), \tag{2.50}$$

$$P(K;\Pi) = \prod_{t=1}^{T} \prod_{n=1}^{N_t} \pi_t^{k_{nt}}, \tag{2.51}$$

$$P(Z|X,K) = \prod_{t=1}^{T} \prod_{n=1}^{N_t} \zeta\left(z_{nt}|x_t^{k_{nt}}\right). \tag{2.52}$$

The conditional probability of the missing data, $P(K|\hat{X}(i), Z; \hat{\Pi})$, can be determined using Bayes' Rule:

$$\begin{aligned}
P(K|\hat{X}, Z; \hat{\Pi}) &= \frac{P(\hat{X}, K, Z; \hat{\Pi})}{P(\hat{X}, Z; \hat{\Pi})} \\
&= \frac{P(\hat{X}, K, Z; \hat{\Pi})}{\sum_K P(\hat{X}, K, Z; \hat{\Pi})} \\
&= \frac{P(\hat{X})P(K; \hat{\Pi})P(Z|\hat{X}, K)}{P(\hat{X})\sum_K P(K; \hat{\Pi})P(Z|\hat{X}, K)} \\
&= \frac{P(K; \hat{\Pi})P(Z|\hat{X}, K)}{\sum_K P(K; \hat{\Pi})P(Z|\hat{X}, K)} \\
&= \frac{\prod_{t=1}^{T}\prod_{n=1}^{N_t} \pi_t^{k_{nt}} \zeta(z_{nt}|x_t^{k_{nt}})}{\sum_K \prod_{t=1}^{T}\prod_{n=1}^{N_t} \pi_t^{k_{nt}} \zeta(z_{nt}|x_t^{k_{nt}})} \\
&= \prod_{t=1}^{T}\prod_{n=1}^{N_t} \frac{\pi_t^{k_{nt}} \zeta(z_{nt}|x_t^{k_{nt}})}{\sum_{m=0}^{M} \pi_t^m \zeta(z_{nt}|x_t^m)} \\
&\equiv \prod_{t=1}^{T}\prod_{n=1}^{N_t} w_{ntk_{nt}}
\end{aligned} \tag{2.53}$$

where the iteration index (i) is suppressed for clarity.

Thus the conditional probability of the assignments is given by the product of individual per measurement *weights*. Each weight, w_{ntm}, is the normalised likelihood that the nth measurement at time t originated from target m. The numerator of the weight is simply the product of the assignment prior and the measurement likelihood. The weight uses the current state estimate and the current π estimate.

For compactness, let

$$\sum_{n,t,m}(\cdot) \equiv \sum_{t=1}^{T}\sum_{m=0}^{M}\sum_{n=1}^{N_t}(\cdot), \tag{2.54}$$

i.e. a sum over all of the measurements at each time and for each target.

Substituting equations (2.49) and (2.53) into (2.48) leads to the auxiliary function to be maximised:

$$\begin{aligned} Q(X,\Pi|\hat{X}(i),\hat{\Pi}(i)) &= \log P(X) + \sum_{n,t,m} w_{ntm}\log\pi_t^m + \sum_{n,t,m} w_{ntm}\log\zeta(z_{nt}|x_t^m) \\ &\equiv Q_X + Q_\Pi. \end{aligned} \tag{2.55}$$

The term Q_Π in (2.55) is given by

$$Q_\Pi \equiv \sum_{n,t,m} w_{ntm}\log\pi_t^m.$$

It is maximised subject to the constraint

$$\sum_{m=0}^{M}\pi_t^m = 1, \tag{2.56}$$

meaning that Π_t should be a proper probability vector. This is achieved by using a Lagrangian

$$L \equiv Q_\Pi + \sum_{t=1}^{T}\gamma_t(1-\sum_{m=0}^{M}\pi_t^m), \tag{2.57}$$

where γ_t is the Lagrange multiplier. Differentiating the Lagrangian with respect to π_t^m and

setting the result to zero gives the necessary condition

$$\pi_t^m = \frac{1}{\gamma_t} \sum_{n=1}^{N_t} w_{ntm}. \tag{2.58}$$

Summing these equations from $m = 0$ to M and using the constraint 2.56 gives

$$\gamma_t = \sum_{n=1}^{N_t} \sum_{m=0}^{M} w_{ntm}, \tag{2.59}$$

resulting in the updated prior estimate

$$\hat{\pi}_t^m(i+1) = \frac{1}{N_t} \sum_{n=1}^{N_t} w_{ntm}, \tag{2.60}$$

i.e. the weights' relative frequency for time t.

The remaining term, Q_X, couples the target states and the measurements and is given by

$$Q_X \equiv \log P(X) + \sum_{n,t,m} w_{ntm} \log \zeta(z_{nt}|x_t^m). \tag{2.61}$$

Intuitively, if $\zeta(z_{nt}|x_t^m)$ is Gaussian, then the summation in (2.61) is a linear combination of quadratics, which is itself a quadratic. This allows us to write (2.61) as

$$Q_X \equiv \log P(X) + \sum_{t,m} \log \tilde{\zeta}_t(\tilde{z}_t^m|x_t^m), \tag{2.62}$$

where the synthetic measurement, $\tilde{z}_t^m$, is given by

$$\tilde{z}_t^m = \frac{1}{\sum_{n=1}^{N_t} w_{ntm}} \sum_{n=1}^{N_t} w_{ntm} z_{nt}, \tag{2.63}$$

and the synthetic measurement function, $\tilde{\zeta}_t(\cdot)$, is a Gaussian pdf with the same mean as the true measurement function and a variance that is a scaled version of the sensor measurement variance, R_t^m,

$$\tilde{R}_t^m = \frac{1}{\sum_{n=1}^{N_t} w_{ntm}} R_t^m. \tag{2.64}$$

Algorithm 1 PMHT algorithm

1: Initialise target state estimates and uniform measurement-to-track assignment priors.
2: Calculate the assignment weight for each measurement and track at each scan, according to (2.53).
3: Update the assignment priors using (2.60).
4: Determine synthetic measurements and covariances for each target at each scan using (2.63) and (2.64).
5: Update the target state estimates using a Kalman smoothing algorithm over the synthetic measurements and covariances.
6: repeat steps 2 to 5 until convergence of the auxilliary function (2.55).

The track for target m is now refined by smoothing the synthetic measurements and covariances.

The PMHT consists of iteratively calculating assignment weights, w_{ntm}, and estimating the target tracks and assignment priors until convergence. For a linear Gaussian problem this amounts to the process detailed in Algorithm 1.

Chapter 3

PMHT with time uncertainty

3.1 Introduction

The role of a tracking filter is to perform recursive target state estimation given a set of noisy position measurements collected at known times. It is customary to assume that there is no error in the timing information, and this assumption is reasonable for the case of a single electromagnetic sensor. In such a case, the propagation delay from sensor to target and back is negligible and the tracker is situated at the sensor, so it knows the sensor clock. However, for other situations, such as distributed sensing, the timing information may not be so reliable. As the trend for sensor networks heads towards the usage of low cost sensors such as motes and other low bandwidth devices, this time uncertainty error will need to be accounted for.

Relatively little work has been done in the area of estimation with uncertain timing information. Li and Leung considered a fixed timing bias in the context of sensor registration [LL06]. Their algorithm used Expectation Maximisation [DLR77] to estimate time, range and azimuth bias parameters.

Morelande [Mor08] considered a problem where some measurements have accurate time information, but others do not. Morelande used the accurate time information to estimate the parameters of the unreliable time-stamps via a particle filter. This problem will be explored more thoroughly as an example of the PMHT-t described in this chapter.

Orguner and Gustafsson [OG08, OG09] considered a problem where the propagation delay was significant and the received time was significantly different to the time the sensor energy was reflected from the target. This propagation delay was treated as a state dependent time bias.

Morelande's and Orguner's methods extended the state vector to estimate the delay and hence improve tracking.

This chapter proposes an extension to the PMHT algorithm to track using measurements with time errors. The PMHT algorithm derived in this chapter uses EM to estimate target trajectories when the measurement time is not known reliably. This extension of the PMHT is referred to as the PMHT-t. The derivation of the PMHT-t follows the same development as the original PMHT algorithm and is presented in full here for completeness.

The PMHT performs data association by introducing an assignment index that is an unobserved random variable and then marginalising over it via EM. This chapter extends this model by introducing another assignment index that identifies the true time at which a measurement was collected. Again, this index is unobserved and the EM method is used to marginalise over it. The result is a set of weights that probabilistically associate each measurement with both tracks and time slots. It is shown that the resulting EM auxiliary function is equivalent to a problem with known observation times and known measurement origin using synthetic measurements and a modified measurement function.

This chapter is organised as follows. The first section formulates a generic time uncertainty problem for multiple targets and measurements. Section 3.3 derives the PMHT-t algorithm to solve the time uncertainty problem under the assumption that the statistical distribution of the timing noise is known. This is followed by a comparison of PMHT-t with PMHT for a tracking example with a manoeuvering target and delayed time measurements. Section 3.5 considers the scenario from [Mor08] where some measurements have accurate time information, but others do not. The PMHT-t is extended to the case where the parametric form of the timing noise is known, but its parameters are not. PMHT-t is used to estimate these parameters and its performance is compared with the alternative algorithms used in [Mor08].

3.2 Problem Formulation

Assume that there is a single sensor that is tracking M targets. At discrete scans, it collects observations, some of which are due to the targets, and some of which are false alarms, referred to as clutter. Let the total number of observations be N and the number of scans be T.

As with the formulation in Chapter 2, let the state of target m at time t be denoted x_t^m, and let X^m denote the set of all states for target m, i.e. $X^m = \{x_t^m\}$ for $t = 1 \dots T$. Similarly, let the set of all states across the targets be denoted by $X = \{X^m\}$.

It is assumed that the prior distribution of the state of each target is known and is given by $\psi_0^m(x_0^m)$ for target m. The target dynamics are also assumed to be known and can be described by the evolution pdf $\psi_t^m(x_t^m|x_{t-1}^m)$.

Let the nth measurement received by the sensor be denoted by z_n. Each measurement consists of a state component, z_n^x, and a time stamp, z_n^τ, both of which are noisy observations.

Let the true source of measurement n be $k_n \in \{0 \dots M\}$, where $k_n = 0$ means that the measurement is a false alarm and $k_n = m$ implies that measurement n is an observation of target m. Let the true collection time of measurement n be $\tau_n \in \{1 \dots T\}$. Both of these are, of course, unknown and are treated as independent identically distributed random variables with priors π_{nm}^k and π_{nt}^τ respectively. These prior distributions are also unknown.

Let Z denote the set of all measurements, K denote the set of all measurement-to-platform assignments and τ denote the set of all measurement-to-time assignments. Similarly, Π^k and Π^τ denote the collection of track and time assignment priors respectively.

The sensor observation process is described by a known measurement pdf that is denoted as $\zeta^x(z_n^x|x_t^m)$, where $m = k_n$ and $t = \tau_n$. In general PMHT is capable of handling a time and target dependent measurement function, but here we choose to assume a constant one for notational clarity.

Similarly, the time stamp has a conditional probability mass function (pmf) that is denoted as $\zeta^\tau(z_n^\tau|\tau_n)$. This pmf may be modelled if known and estimated if unknown, as shown in the results in Sections 3.4 and 3.5. Note that the time has been assumed to be discrete whereas in general, sensors may collect data at arbitrary times. It is possible to undertake the algorithm development in continuous time, however, any implementation would require time discretisation. For this reason, the algorithm is derived with discrete time, with the caveat that the sample rate is a design choice and can be changed if required.

3.3 PMHT for tracking with timing uncertainty

For clarity, the abbreviation PMHT will be used to refer to the standard PMHT algorithm, i.e. tracking assuming perfect timing information. The abbreviation PMHT-t will be used to refer to the modified PMHT that is capable of compensating for unreliable timing information. The derivation of the PMHT-t is similar to the standard PMHT derivation presented in detail in chapter 2; the difference is the inclusion of the extra hidden variable τ and the extended measurement function $\zeta^\tau(z_n^\tau|\tau_n)$.

In EM terminology, the complete data are (X, τ, K, Z), the incomplete data are (X, Z), and (τ, K) are the missing data. The auxiliary function is the expectation of the complete data log-likelihood over the missing data, which now takes the form:

$$Q(X, \Pi^\tau, \Pi^k | \hat{X}(i), \hat{\Pi}^\tau(i), \hat{\Pi}^k(i)) = \sum_K \sum_\tau P(\tau, K | \hat{X}(i), Z; \hat{\Pi}^\tau(i), \hat{\Pi}^k(i)) \log P(X, \tau, K, Z; \Pi^\tau, \Pi^k), \quad (3.1)$$

where the summation is over all permutations of the assignment variables τ and K, and $\hat{X}(i)$ denotes an estimated variable on the ith EM iteration.

Due to the independence assumptions, the complete data likelihood becomes:

$$P(X, \tau, K, Z; \Pi^\tau, \Pi^k) = P(X) P(\tau; \Pi^\tau) P(K; \Pi^k) P(Z | X, \tau, K), \quad (3.2)$$

where

$$P(X) = \prod_{m=1}^{M} \psi_0^m(x_0^m) \prod_{t=1}^{T} \psi_t^m(x_t^m | x_{t-1}^m), \quad (3.3)$$

$$P(\tau; \Pi^\tau) = \prod_{n=1}^{N} \pi_{n\tau_n}^\tau, \quad (3.4)$$

$$P(K; \Pi^k) = \prod_{n=1}^{N} \pi_{nk_n}^k, \quad (3.5)$$

$$P(Z | X, \tau, K) = \prod_{n=1}^{N} \zeta^x \left(z_n^x | x_{\tau_n}^{k_n} \right) \zeta^\tau \left(z_n^\tau | \tau_n \right) \quad (3.6)$$

The main difference from the standard PMHT model is the inclusion of the extended measurement function $\zeta^\tau \left(z_n^\tau | \tau_n \right)$ with the standard positional pdf to give the overall likelihood in equation (3.6).

The conditional probability of the missing data, $P(\tau, K | \hat{X}(i), Z; \hat{\Pi}^\tau(i), \hat{\Pi}^k(i))$ in equation

(3.1), can be determined using Bayes' Rule:

$$
\begin{aligned}
&P(\tau, K|\hat{X}, Z; \hat{\Pi}^{\tau}, \hat{\Pi}^{k}) \\
&\quad = \frac{P(\hat{X}, \tau, K, Z; \hat{\Pi}^{\tau}, \hat{\Pi}^{k})}{\sum_{\tau,K} P(\hat{X}, \tau, K, Z; \hat{\Pi}^{\tau}, \hat{\Pi}^{k}))} \\
&\quad = \frac{P(\hat{X})P(\tau; \hat{\Pi}^{\tau})P(K; \hat{\Pi}^{k})P(Z|\hat{X}, \tau, K)}{P(\hat{X})\sum_{\tau,K} P(\tau; \hat{\Pi}^{\tau})P(K; \hat{\Pi}^{k})P(Z|\hat{X}, \tau, K)} \\
&\quad = \frac{\prod_{n} \pi^{\tau}_{n\tau_n}\pi^{k}_{nk_n}\zeta^{x}(z^{x}_{n}|x^{k_n}_{\tau_n})\zeta^{\tau}(z^{\tau}_{n}|\tau_n)}{\sum_{\tau',K'}\prod_{n} \pi^{\tau}_{n\tau'_n}\pi^{k}_{nk'_n}\zeta^{x}(z_{n}|x^{k'_n}_{\tau'_n})\zeta^{\tau}(z^{\tau}_{n}|\tau'_n)} \\
&\quad = \prod_{n} \frac{\pi^{\tau}_{n\tau_n}\pi^{k}_{nk_n}\zeta^{x}(z^{x}_{n}|x^{k_n}_{\tau_n})\zeta^{\tau}(z^{\tau}_{n}|\tau_n)}{\sum_{t=1}^{T}\sum_{m=0}^{M} \pi^{\tau}_{nt}\pi^{k}_{nm}\zeta^{x}(z^{x}_{n}|x^{m}_{t})\zeta^{\tau}(z^{\tau}_{n}|t)} \\
&\quad \equiv \prod_{n} w_{n\tau_n k_n} \qquad (3.7)
\end{aligned}
$$

where the iteration (i) is suppressed for clarity.

Once again, the main difference between this conditional probability and that of the standard PMHT is the inclusion of the pmf of the timing error.

Thus the conditional probability of the assignments is given by the product of individual per measurement *weights*. Each weight, w_{ntm}, is the normalised likelihood of the nth measurement being from target m at time t. The numerator of the weight is the product of the assignment priors, the positional measurement likelihood and the temporal measurement likelihood.

Substituting equations (3.2) and (3.7) into (3.1) leads to the auxiliary function to be maximised:

$$
\begin{aligned}
&Q(X, \Pi^\tau, \Pi^k | \hat{X}(i), \hat{\Pi}^\tau(i), \hat{\Pi}^k(i)) \\
&= \sum_K \sum_\tau P(\tau, K|X, Z) \log \left(P(X) P(\tau; \Pi^\tau) P(K; \Pi^k) P(Z|X, \tau, K) \right) \\
&= \log P(X) + \sum_{n,t,m} w_{ntm} \log \pi^\tau_{nt} + \sum_{n,t,m} w_{ntm} \log \pi^k_{nm} \\
&\quad + \sum_{n,t,m} w_{ntm} \log \zeta^x(z^x_n | x^m_t) + \sum_{n,t,m} w_{ntm} \log \zeta^\tau(z^\tau_n | t) \\
&\equiv Q_X + Q^\tau_\Pi + Q^k_\Pi + Q_\tau
\end{aligned} \tag{3.8}
$$

The term Q_τ in (3.8) is given by

$$Q_\tau = \sum_{n,t,m} w_{ntm} \log \zeta^\tau(z^\tau_n | t)$$

and is constant so it has no influence on the optimisation.

The term Q^τ_Π in (3.8) is given by

$$Q^\tau_\Pi \equiv \sum_{n,t,m} w_{ntm} \log \pi^\tau_{nt}.$$

It is maximised subject to the constraint that $\sum_t \pi^\tau_{nt} = 1$ by a Lagrangian in the same way as Q_π in the standard PMHT detailed in Section 2.3. The resulting updated prior estimate is

$$\hat{\pi}^\tau_{nt}(i+1) = \sum_{m=0}^{M} w_{ntm}, \tag{3.9}$$

i.e. the weights' relative frequency for time t.

Similarly, the Q^k_Π term results in a relative frequency estimate for the locale to platform assignment prior

$$\hat{\pi}^k_{nm}(i+1) = \sum_{t=1}^{T} w_{ntm}. \tag{3.10}$$

The remaining term, Q_X, couples the target states and the measurements and is given by

$$Q_X \equiv \log P(X) + \sum_{n,t,m} w_{ntm} \log \zeta^x(z_n|x_t^m). \tag{3.11}$$

This function is almost the same as that for state estimation in the standard PMHT. However, in this case, every measurement potentially contributes to state estimation at every time step, as dictated by the weights. The assignment weight for measurement n at time step t depends on a combination of the physical distance between the target state at t and the difference between t and the observed time z_n^τ where the relative importance of each is dictated by the measurement noise covariance R and the time measurement model. Effectively the PMHT-t has the freedom to trade spatial distance for temporal distance. This can have the surprising result that positional estimation accuracy can be improved in some cases at the expense of temporal estimation accuracy, that is, the best-case RMS position error may not happen when all of the measurements are assigned to the correct time steps.

As before, if the measurement function is Gaussian, it can be shown that this function is equivalent to the log likelihood of a tracking problem with known data association [Dav07]. Intuitively, if $\zeta^x(z_n|x_t^m)$ is Gaussian, then the summation in (3.11) is a linear combination of quadratics, which is itself a quadratic. This allows us to write (3.11) as

$$Q_X \equiv \log P(X) + \sum_{t,m} \log \tilde{\zeta}_t^x(\tilde{z}_t^m|x_t^m), \tag{3.12}$$

where the synthetic measurement, $\tilde{z}_t^m$, is given by

$$\tilde{z}_t^m = \frac{1}{\sum_{n=1}^N w_{ntm}} \sum_{n=1}^N w_{ntm} z_n, \tag{3.13}$$

and the synthetic measurement function, $\tilde{\zeta}(\cdot)$, is a Gaussian pdf with the same mean as the true measurement function and a variance that is a scaled version of the sensor measurement variance, R,

$$\tilde{R}_t^m = \frac{1}{\sum_{n=1}^N w_{ntm}} R. \tag{3.14}$$

The track for target m is now refined by smoothing the synthetic measurements and covariances.

Algorithm 2 PMHT-t algorithm

1: Initialise target state estimates, measurement-to-track assignment priors and measurement-to-time assignment priors.
2: Calculate the assignment weight for each measurement and track at each scan, according to (3.7).
3: Update the assignment priors using (3.9) and (3.10).
4: Determine synthetic measurements and covariances for each target at each scan using (3.13) and (3.14).
5: Update the target state estimates using a Kalman smoothing algorithm over the synthetic measurements and covariances.
6: repeat steps 2 to 5 until convergence of the auxiliary function (3.8).

The PMHT-t consists of iteratively calculating assignment weights, w_{ntm}, and estimating the target tracks and assignment priors until convergence. For a linear Gaussian problem this amounts to the procedure in Algorithm 2. A listing of Matlab source code for the PMHT-t algorithm is in Appendix A.1.

3.4 Performance Analysis

The use of PMHT for tracking with time uncertainty is now illustrated through simulation. The simulations assumed a Cartesian sensor observing targets in the plane. The moving targets had a state vector consisting of 2-D position and velocity. For each scenario 100 scans were simulated.

The target states were assumed to follow a constant velocity model independently in X and Y, as described in Section 2.2.2.

The time measurement model of the sensor was a stochastic delay: a measurement could not be received before the true collection time, i.e. $\tau_n \leq z_n^\tau$, and the probability delayed exponentially:

$$\zeta^\tau(z_n^\tau|\tau_n) = \begin{cases} 0 & z_n^\tau < \tau_n, \\ p(1-p)^{(z_n^\tau - \tau_n)} & z_n^\tau > \tau_n. \end{cases} \tag{3.15}$$

Thus zero delay occurred with probability p and the mean delay was $(1-p)/p$. Figure 3.1 shows the time error probability mass function for various values of p versus time delay.

The sensor was assumed to have a constant probability of detection, P_d. The number of false alarms was Poisson distributed with a mean of γ false alarms per scan, and the false alarms were

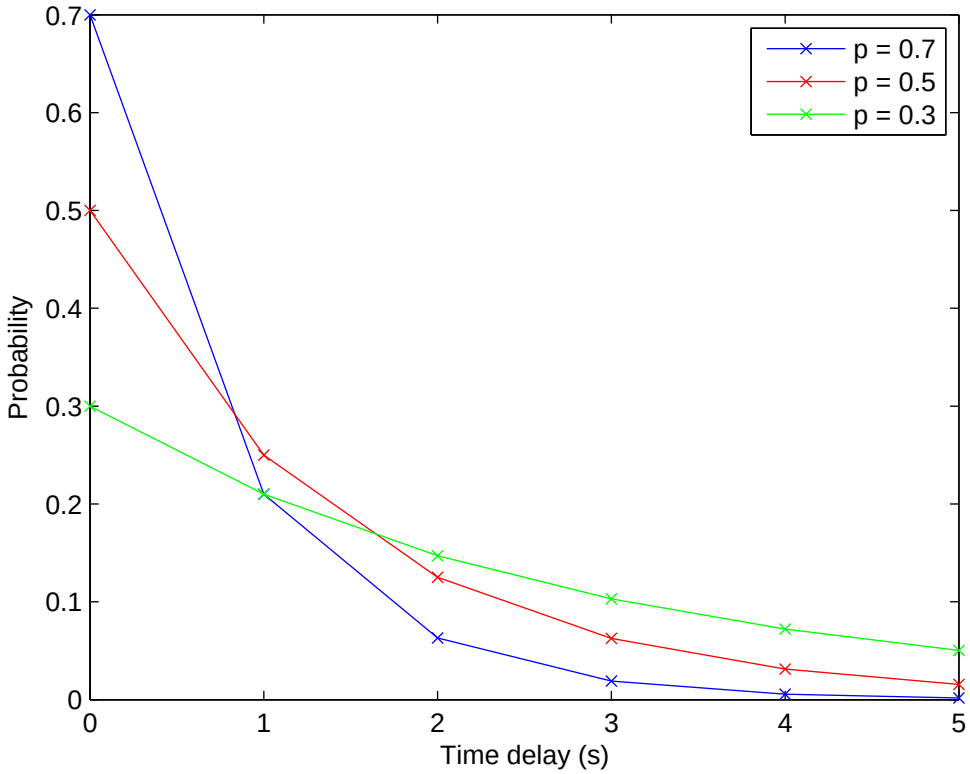


Figure 3.1: Time error pmf

uniformly distributed in space.

The measurement-to-track assignment prior was initialised with

$$\hat{\pi}^x_{nm}(0) = \begin{cases} \gamma + \epsilon & m = 0, \\ P_d & m > 0, \end{cases} \tag{3.16}$$

where ϵ is a small constant used to ensure good conditioning for very low clutter densities. This initialisation is not normalised, but due to the ratio in the weight calculation (3.7), the scale factor cancels out and has no effect.

A sliding batch with $T = 10$ was run over the measurements and the measurement-to-time assignment prior was initialised with

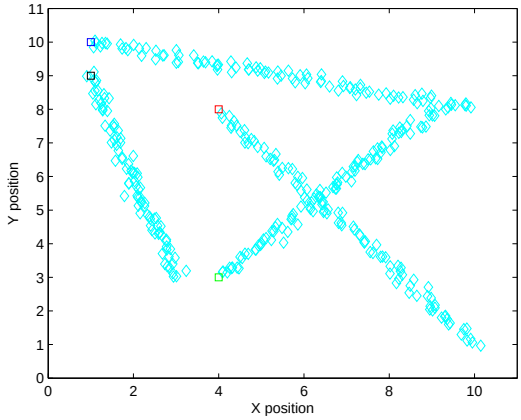
$$\hat{\pi}^\tau_{nt}(0) = \frac{1}{T}. \tag{3.17}$$

Two measures of performance were used to quantify the quality of the track output. Firstly, the fraction of misassociated tracks was determined, where a missassociated track was defined as one that did not remain within a prescribed circular gate of the true target location for the whole scenario duration. For example, a track that swaps from one target to another was classed as misassociated, or one that diverged from the target path due to clutter. Secondly, the 2-D RMS position and velocity estimation errors were determined. Misassociated tracks were not included in the error calculation.

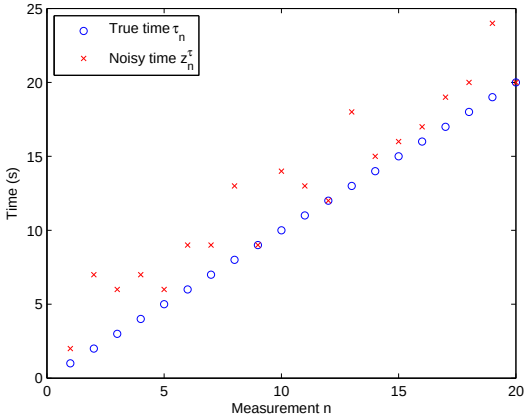
The first two scenarios are presented here with no clutter. In the first scenario, there were four targets travelling with constant-velocity motion as illustrated in Figure 3.2. Figure 3.2(a) shows a single realisation of target measurements with a P_d of 0.9 and no clutter ($\gamma = 0$). The coloured boxes show the target initial locations. Two of the targets had clear paths, not close to the others, but the remaining two targets crossed part way through the scenario. Figure 3.2(b) shows a comparison of the first twenty noisy time measurements to the true times. In this example, it can be seen that the noisy measurements may be delayed up to 5 seconds.

Figures 3.2(c) and (d) show the tracks resulting from running the PMHT-t and the standard PMHT respectively. In this particular example, due to the timing uncertainty, the standard PMHT has misassociated the crossing targets whereas the PMHT-t has tracked the targets correctly.

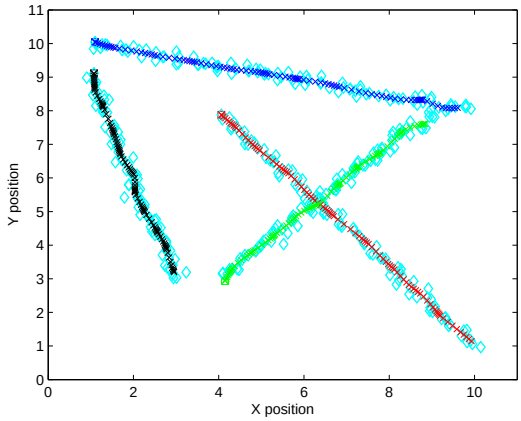
The second simulation scenario involved a single target performing a manoeuvre. The target



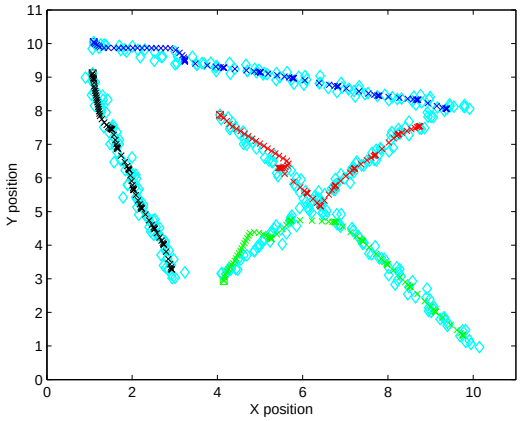
(a) Simulated measurements



(b) Time measurements



(c) PMHT-t results



(d) PMHT results

Figure 3.2: Scenario 1 Tracking Results

Table 3.1: Scenario 1 Monte Carlo RMS results

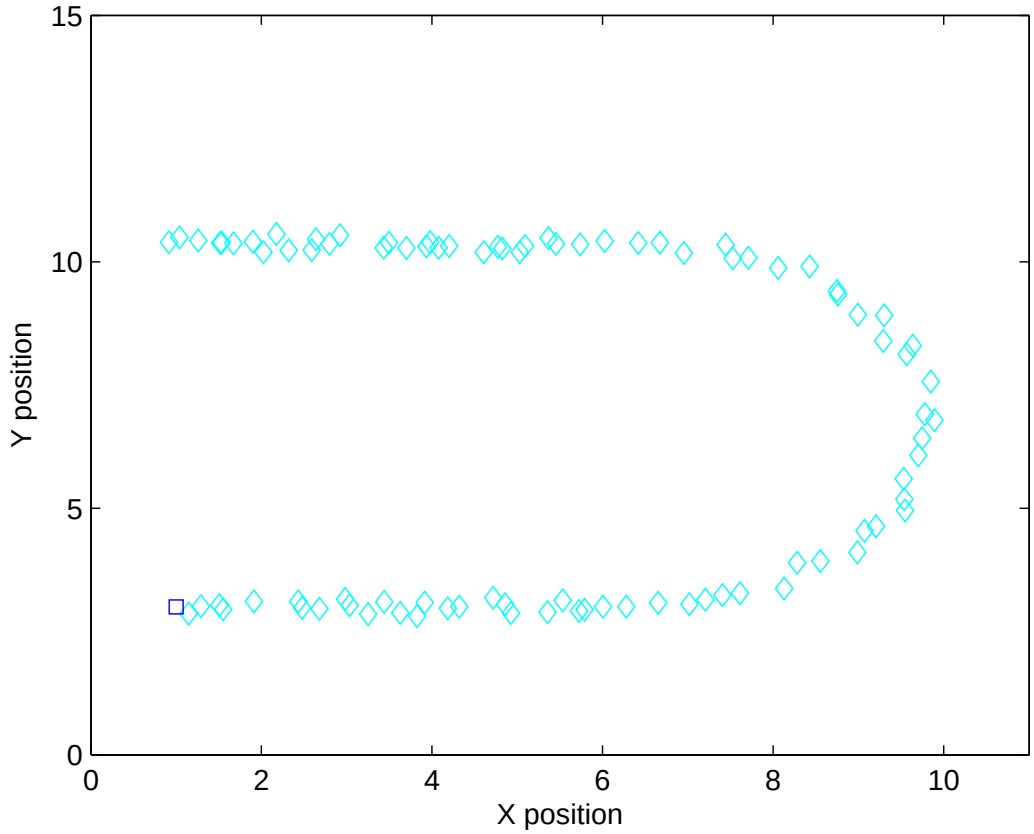
p	1.0	0.7	0.5	0.3
av. delay	0	0.4	1	2.3
Position RMS				
PMHT	0.09	0.21	0.16	0.27
PMHT-t	0.09	0.11	0.14	0.23
PMHT-t $(p' = 0.7)$	0.09	0.11	0.14	0.23
Velocity RMS				
PMHT	0.02	0.03	0.04	0.07
PMHT-t	0.02	0.03	0.03	0.05
PMHT-t $(p' = 0.7)$	0.02	0.03	0.03	0.05
Misassociated				
PMHT	0	0.01	0.03	0.06
PMHT-t	0	0.01	0.02	0.04
PMHT-t $(p' = 0.7)$	0	0.01	0.02	0.04

initially travelled East with constant velocity before performing a coordinated turn manoeuvre with constant turn rate and then travelling West, again with constant velocity. Figure 3.3(a) shows a single realisation of target measurements with a P_d of 0.9 and no clutter ($\gamma = 0$).

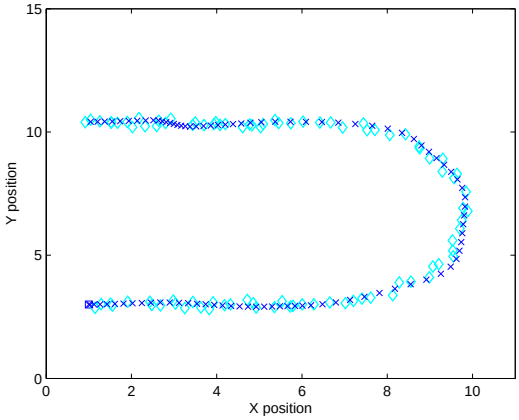
Figures 3.3(b) and (c) show the results from running PMHT-t and the standard PMHT respectively. In this particular case, the standard PMHT was not able to follow the manoeuvre due to the time errors and broke track whereas the PMHT-t was able to track the target.

Monte Carlo simulations of the above scenarios were performed, and the fraction of misassociated tracks and RMS in position and velocity was averaged over 100 trials for different timing delay parameters, p. The results for scenario one and two are in Table 3.1 and Table 3.2 respectively. In order to test the sensitivity of the PMHT-t to mismatch in the timing delay pmf, two versions of PMHT-t were tested. In one, a fixed value of $p' = 0.7$ was used, while the true p used to generate data was varied. In the second, the PMHT-t was provided with the true parameter value.

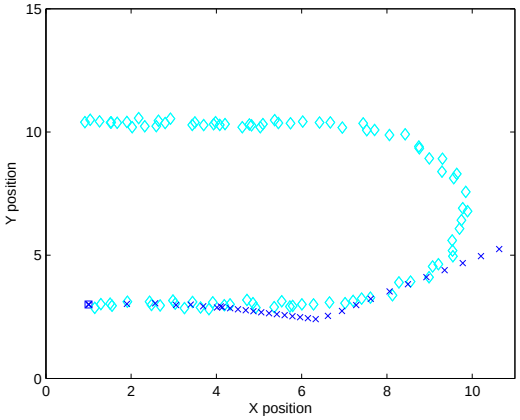
Both Table 3.1 and Table 3.2 show that in the case where $p = 1$, the PMHT and the PMHT-t gave the same results. In this case, there was no timing error. As the value of p decreased, there was a higher probability of time delay error in the measurements and the targets were more difficult to track. Table 3.1 shows that the RMS error for the standard PMHT was slightly greater than that of the PMHT-t. There were also less misassociated tracks with the PMHT-t.



(a) Simulated measurements



(b) PMHT-t results



(c) PMHT results

Figure 3.3: Scenario 2 Tracking Results

Table 3.2: Scenario 2 Monte Carlo RMS results

p	1.0	0.7	0.5	0.3
av. delay	0	0.4	1	2.3
Position RMS				
PMHT	0.34	0.73	0.86	1.18
PMHT-t	0.34	0.24	0.35	0.57
PMHT-t ($p' = 0.7$)	0.17	0.24	0.35	0.56
Velocity RMS				
PMHT	0.08	0.22	0.25	0.31
PMHT-t	0.08	0.06	0.07	0.09
PMHT-t ($p' = 0.7$)	0.05	0.06	0.07	0.09
Misassociated				
PMHT	0	0.32	0.41	0.51
PMHT-t	0	0	0	0
PMHT-t ($p' = 0.7$)	0	0	0	0

For the straight-line scenario, there was very little difference in performance. This was not the case for the turn scenario, as shown in Table 3.2. In this scenario, when there was significant timing error, the standard PMHT had much more difficulty following the target manoeuvre. This is evident both in the RMS error and the number of misassociations. In contrast, the PMHT-t managed to track the target through the manoeuver throughout all the trials.

Interestingly, in the case where $p = 1$, the PMHT-t with an incorrectly assumed $p' = 0.7$ was found to have a lower RMS error on the turning scenario. In this case, the algorithm has a mechanism for dealing with outlier measurements that reduces their impact on the tracking performance. In this example an outlier measurement is one with a high amount of spatial measurement noise. The standard PMHT and the PMHT-t that correctly uses $p = 1$ know the true time that this outlier was collected and therefore must make a large correction to the state estimate at this time. In contrast, the mismatched PMHT-t may find an incorrect time point that has a track position with a smaller spatial error to the measurement and associate the measurement to this time instead of the true time. As described in Section 3.3 by the Q_X term, the mismatched PMHT-t has the freedom to trade spatial error for temporal error. In this case, by assigning measurements to the wrong time, the variance of the error between the true target state and the time-shifted measurements is less than R. This must be the case since any individual measurement will only be assigned to the wrong time if it is spatially closer. Since

the time-shifted measurements have a smaller effective measurement variance, the resulting state estimate will be more accurate. This especially applies to measurements where the spatial error happens to lie longitudinally along the true path of the target.

When there was non-zero timing error, the mismatched PMHT-t gave almost identical performance to the matched PMHT-t. This indicates that the algorithm is robust to errors in the timing measurement function. It appears that the precise shape of the assumed time measurement pmf is not highly important provided that it gives support over the region where the true function has significant mass.

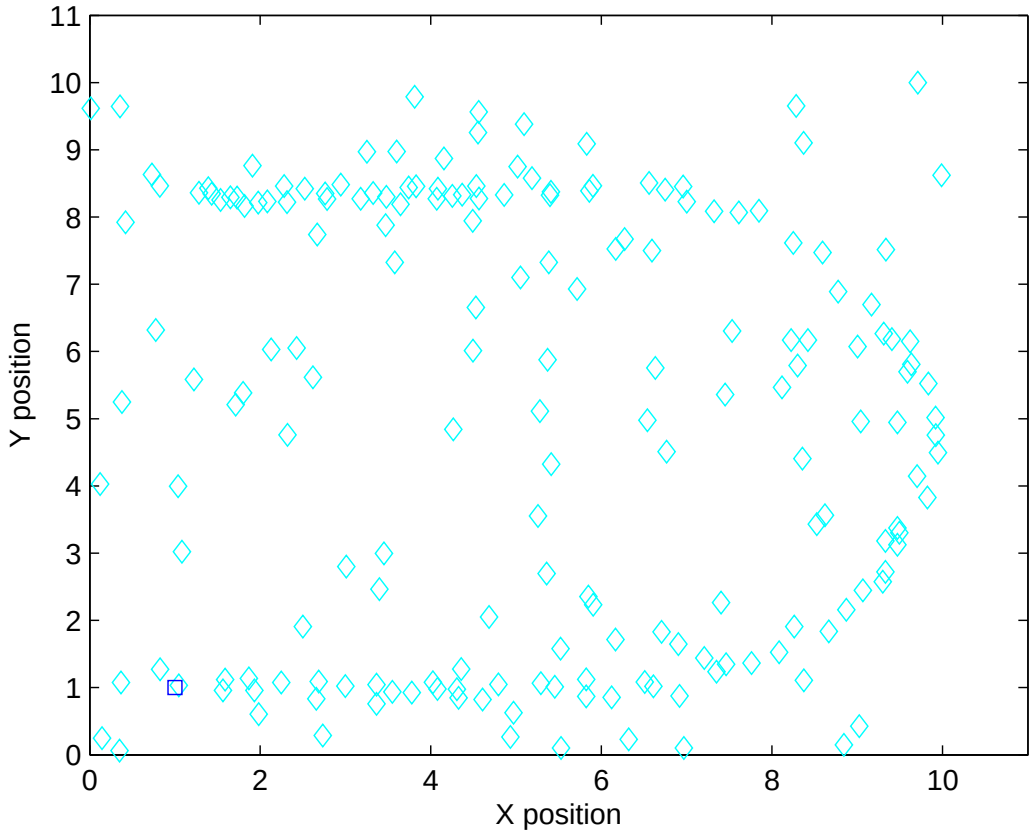
The performance of the two algorithms was also compared for scenario 2, the manoeuvering target trajectory, with clutter measurements included. Figure 3.4(a) shows a sample realisation of the scenario for a P_d of 0.9 and clutter density of $\gamma = 1$. Figures 3.4(a) and 3.4(b) show the results from running PMHT-t and the standard PMHT respectively.

Monte Carlo simulations of this scenario were performed similar to the first two scenarios, with the results shown in Table 3.3. Compared to the scenarios with no clutter, the number of misassociated tracks has increased quite considerably. However, the number of misassociated tracks with the PMHT-t remains considerably less than with the standard PMHT, with an improvement of up to 35%. Similar to before, the PMHT-t with an incorrectly assumed value of $p' = 0.7$ is observed to perform better than the other two algorithms. Interestingly, in the scenario with no time delay errors $p = 1.0$, the PMHT-t with the $p' = 0.7$ has half the number of misassociated tracks compared with the PMHT-t. There were also minor improvements in the position and velocity RMS errors.

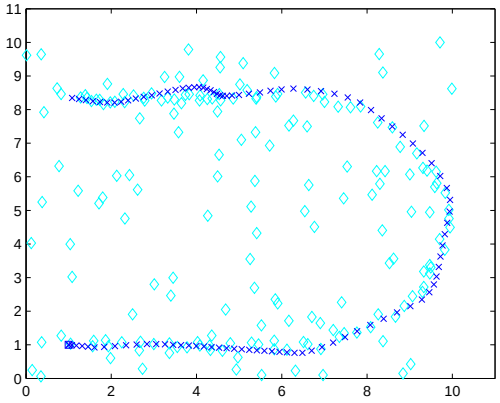
3.5 Comparison of PMHT-t with other timing uncertainty approaches

The PMHT-t is now compared with other algorithms that are capable of resolving time uncertainty based on the work of Morelande [Mor08]. Morelande considered a large-scale sensor network composed of inexpensive sensor nodes with limited resources and proposed a scenario where it was possible to receive unreliable timing due to communication failures or synchronisation errors.

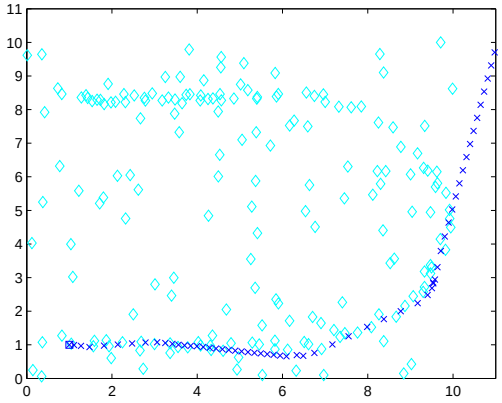
In the scenario, the measurements collected by the sensor were communicated to a processing node that formed tracks. Although the processing node knew the time of receipt of the



(a) Simulated measurements



(b) PMHT-t results



(c) PMHT results

Figure 3.4: Scenario 2 with Clutter Tracking Results

Table 3.3: Scenario 2 with clutter Monte Carlo RMS results

p	1.0	0.7	0.5	0.3
av. delay	0	0.4	1	2.3
Position RMS				
PMHT	0.72	1.19	1.40	1.36
PMHT-t	0.78	0.75	0.89	0.86
PMHT-t $(p' = 0.7)$	0.57	0.75	0.84	0.98
Velocity RMS				
PMHT	0.27	0.40	0.45	0.40
PMHT-t	0.31	0.29	0.38	0.38
PMHT-t $(p' = 0.7)$	0.21	0.29	0.38	0.40
Misassociated				
PMHT	0.37	0.88	0.93	0.9
PMHT-t	0.37	0.27	0.43	0.56
PMHT-t $(p' = 0.7)$	0.16	0.27	0.4	0.55

communication, there was a stochastic transmission delay. The sensor time stamped its data but this time stamp was error prone and was assumed to be available for only a subset of the data. In practice, the processing node may receive data from many sensors, some of which have negligible time stamp errors and some of which do not. The aim of this tracking problem is to both track the target state using the measurements, and to estimate the statistics of the stochastic transmission delay in order to correct the time stamps when the true collection time is unavailable.

It was assumed that the functional form of the timing error was known but that its parameters were unknown. The timing error parameters were added to the state vector as an augmented state vector. [Mor08] uses three algorithms to approximate the optimal solution. The first algorithm computes the target track and the parameter estimates only when the true measurement times are available. The second algorithm replaces unavailable measurement times with a point estimate obtained from the noisy measurement time and the current estimates of the timing error parameters. The third algorithm computes a numerical approximation to the posterior distribution of the augmented state using a sequential Monte Carlo method.

These algorithms are now compared with a version of the PMHT-t that uses the timing error parameters to perform measurement-to-time data association to track the target. The PMHT-t used in this section treats the time error parameters as additional states to be estimated in the

M-step of the EM process.

3.5.1 PMHT-t Solution

The problem is to recursively estimate the state x_t for $t = 1, 2, \ldots, T$ given the measurements $Z = [z_1, ..., z_N]^{\mathsf{T}}$ and their true measurement times $\tau_1, \ldots, \tau_N$. It is assumed the true measurement times τ_n are available only occasionally while noisy measurement times z_n^τ are available at every sampling instant.

The noisy measurement time z_n^τ for the nth measurement is assumed to satisfy

$$\zeta^\tau(z_t^\tau|\tau_n;\mu,\lambda) = N(\tau_n + \mu, 1/\lambda) \tag{3.18}$$

where the stochastic transmission delay between z_n^τ and τ_n is assumed to be Gaussian with unknown parameters, mean μ and inverse variance λ.

The PMHT-t as derived in Section 3.3 is used to solve this problem. Whereas the time stamp pmf, $\zeta^\tau(z_n^\tau|\tau_n)$ was previously known, now it has unknown parameters.

This change results in a slightly different auxiliary function

$$\begin{aligned}
&Q(X, \Pi^\tau, \Pi^k | \hat{X}(i), \hat{\Pi}^\tau(i), \hat{\Pi}^k(i)) \\
&\quad = \sum_K \sum_\tau P(\tau, K|X, Z) \log\left(P(X)P(\tau;\Pi^\tau)P(K;\Pi^k)P(Z|X,\tau,\mu,\lambda,K)\right) \\
&\quad = \log P(X) + \sum_{n,t,m} w_{ntm} \log \pi_{nt}^\tau + \sum_{n,t,m} w_{ntm} \log \pi_{nm}^k \\
&\qquad + \sum_{n,t,m} w_{ntm} \log \zeta^x(z_n^x|x_t^m) + \sum_{n,t,m} w_{ntm} \log \zeta^\tau(z_n^\tau|\tau_n;\mu,\lambda) \\
&\quad \equiv Q_X + Q_\Pi^\tau + Q_\Pi^k + Q_t
\end{aligned} \tag{3.19}$$

where the term Q_t is no longer a constant.

The term Q_t in (3.19) is the only difference compared with (3.8). When the correct measurement time is available, the weights for time measurement n are a delta function. When the correct measurement time is unavailable, their values are as determined by (3.7). Expanding

the term Q_t with respect to whether a correct time measurement is available or not gives

$$\begin{aligned} Q_t &= \sum_{n,t,m} w_{ntm} \log \zeta^{\tau}(z_n^{\tau}|\tau_n;\mu,\lambda) \\ &= \sum_{n\in D}\left\{\frac{1}{2}\log\lambda - \frac{\lambda}{2}(z_n^{\tau}-\tau_n-\mu)^2\right\} + \sum_{n\in\bar{D}}\sum_{t}\sum_{m} w_{ntm}\left\{\frac{1}{2}\log\lambda - \frac{\lambda}{2}(z_n^{\tau}-t-\mu)^2\right\} \end{aligned} \tag{3.20}$$

where D represents when correct time measurements are available and $\bar{D}$ otherwise.

The time error mean μ and variance parameter λ estimates are found by optimising Q_t. Taking the derivative of Q_t and equating to zero gives

$$\frac{\partial Q_t}{\partial \mu} = 0 \tag{3.21}$$

$$\implies 0 = \sum_{n\in D}\lambda(z_n^{\tau}-\tau_n-\mu) + \sum_{n\in\bar{D}}\sum_{t}\sum_{m} w_{ntm}\lambda(z_n^{\tau}-t-\mu) \tag{3.22}$$

$$\mu N = \sum_{n\in D}(z_n^{\tau}-\tau_n) + \sum_{n\in\bar{D}}\sum_{t}\sum_{m} w_{ntm}(z_n^{\tau}-t) \tag{3.23}$$

$$\implies \hat{\mu}(i+1) = \frac{\sum_{n\in D}(z_n^{\tau}-\tau_n) + \sum_{n\in\bar{D}}\left(z_n^{\tau} - \sum_{t}\sum_{m} w_{ntm}t\right)}{N} \tag{3.24}$$

And similarly for the variance,

$$\frac{\partial Q_t}{\partial \lambda} = 0 \tag{3.25}$$

$$\implies 0 = \sum_{n\in D}\left[\frac{1}{2\lambda} - \frac{1}{2}(z_n^\tau - \tau_n - \mu)^2\right] + \sum_{n\in\bar{D}}\sum_t\sum_m w_{ntm}\left\{\frac{1}{2\lambda} - \frac{1}{2}(z_n^\tau - t - \mu)^2\right\} \tag{3.26}$$

$$0 = \frac{N}{2\lambda} - \frac{1}{2}\left[\sum_{n\in D}(z_n^\tau - \tau_n - \mu)^2 + \sum_{n\in\bar{D}}\sum_t\sum_m w_{ntm}(z_n^\tau - \tau_n - \mu)^2\right] \tag{3.27}$$

$$\frac{N}{2\lambda} = \frac{1}{2}\left[\sum_{n\in D}(z_n^\tau - \tau_n - \mu)^2 + \sum_{n\in\bar{D}}\sum_t\sum_m w_{ntm}(z_n^\tau - \tau_n - \mu)^2\right] \tag{3.28}$$

$$\implies \hat{\lambda}(i+1) = \frac{N}{\sum_{n\in D}(z_n^\tau - \tau_n - \mu)^2 + \sum_{n\in\bar{D}}\sum_t\sum_m w_{ntm}(z_n^\tau - t - \mu)^2} \tag{3.29}$$

Equations (3.24) and (3.29) estimate the mean time error and inverse of the variance given the particular output from the PMHT-t algorithm. For the case when the true time is known, the estimate uses the sampled mean and covariance. When the true time is unknown, this solution takes the expectation of the sample mean over the assignment. These parameters converge with each EM iteration to the final estimate.

The PMHT-t consists of iteratively calculating assignment weights, w_{ntm}, and estimating the target tracks, assignment priors and time noise parameters until convergence. For a linear Gaussian problem this amounts to the procedure in Algorithm 3.

3.5.2 Sliding Window PMHT-t

The time error parameter estimation in equations (3.24) and (3.29) assume that the measurements are all processed in a single batch. In a tracking system, it is often desirable to have frequent track updates as the measurements are collected. One technique to achieve this is to use a sliding window. In sliding window PMHT-t, tracking is performed over several batches of measurements rather than the whole set of measurements.

As the time error parameter estimates in (3.24) and (3.29) assume a single batch, these parameter estimates need to be updated with the accumulated data from earlier scans when a sliding window is used. This produces an iterative converging estimate of the time error

Algorithm 3 PMHT-t algorithm with time error estimation

1: Initialise target state estimates, measurement-to-track assignment priors and measurement-to-time assignment priors.
2: Calculate the assignment weight for each measurement and track at each scan, according to (3.7).
3: Update the assignment priors using (3.9) and (3.10).
4: Determine synthetic measurements and covariances for each target at each scan using (3.13) and (3.14).
5: Update the target state estimates using a Kalman smoothing algorithm over the synthetic measurements and covariances.
6: Update time error parameters using equations (3.24) and (3.29).
7: Repeat steps 2 to 6 until convergence of auxiliary function (3.19).

parameters to be used in the PMHT-t.

Assume there are B sliding window batches of measurements and let N_b be the number of measurements in window b = {1,...,B}. Let n_b be the index of the measurements in window b. Now define the mean and variance of the time error parameters based on the measurements from window b to be

$$\hat{\mu}_b = \frac{\sum_{n_b \in D} (z^\tau_{n_b} - \tau_{n_b}) + \sum_{n_b \in \bar{D}} \left(z^\tau_{n_b} - \sum_t \sum_m w_{n_b tm} t \right)}{N_b} \tag{3.30}$$

$$\hat{\lambda}_b = \frac{N_b}{\sum_{n_b \in D} (z^\tau_{n_b} - \tau_{n_b} - \mu)^2 + \sum_{n_b \in \bar{D}} \sum_t \sum_m w_{n_b tm} (z^\tau_{n_b} - t - \mu)^2}. \tag{3.31}$$

Since the time error parameters are calculated once a full window batch of measurements is received, there is no "start up" phase to initialise the sliding window to size n_b.

The updated time error parameters are not only updated with each EM iteration, but also with each batch of measurements. The time error parameters are denoted $\hat{\mu}$ and $\hat{\lambda}$, which are updated using

$$\hat{\mu} \to \frac{N_{b-1}}{N_b} \hat{\mu} + \hat{\mu}_b \tag{3.32}$$

$$\hat{\lambda} \to \frac{N_b}{N_{b-1}} \hat{\lambda} + \hat{\lambda}_b \tag{3.33}$$

Due to this recursive formulation, it is intuitive to see how initial values of μ and λ could be incorporated to shape the algorithm. Higher values of λ will allow larger variance to converge to the correct true time.

These equations estimate the mean time error and variance given the particular output from the PMHT-t algorithm for each sliding window batch and converge with each EM iteration to the final estimate.

3.5.3 Alternative Algorithms

The performance of this PMHT-t based solution is now compared with the alternative algorithms used in [Mor08].

The parameters of the time observation process, μ and λ as described in equation 3.18, are estimated by the process defined by Morelande [Mor08]. When both the true time, τ_n, and the measurement, z_n^τ, are available, the estimates are recursively refined using the equations in (3.34). The initial values of μ_0 can be initialised with 0 and λ an appropriate inverse covariance.

$$\begin{aligned}
\mu_n &= \frac{(n-1)\mu_{n-1} + z_n^\tau - \tau_n}{n} \\
\alpha_n &= n/2 \\
\beta_n &= \beta_{n-1} + \frac{(z_n^\tau - \tau_n - \mu_n)^2(n-1)}{2n} \\
\lambda_n &= \frac{\alpha_n}{\beta_n}
\end{aligned} \tag{3.34}$$

Morelande used the time error estimates with the following different state estimation algorithms:

KF1 - The Kalman filter (as described in Section 2.2.4) using only the measurements with known measurement times.

KF2 - The Kalman filter using measurements with both known and unknown measurement times. The state vector was augmented with a time variable to estimate the parameters of the time delay between the accurate and noisy measurement times. Upon receiving an accurate time measurement, the timing error parameters were updated according to

3.34. When the accurate time was not available, the received time was corrected using the estimated mean error, μ_n.

PF - A numerical approximation to the posterior distribution of the augmented state using a sequential Monte Carlo method, namely a particle filter as described in Section 2.2.7. The particle filter approach was used with a varied number of particle samples: 20, 100 and 500 particle samples, referred to as PF 20, 100 and 500 respectively. The number of particles is a tradeoff of accuracy for computation cost.

3.5.4 Performance Analysis

A scenario containing a single target was simulated with time measurements as described in Section 3.5.1. The target state vector was four dimensional and contained the target position and velocity in a two dimensional Cartesian coordinate system.

The target states were assumed to follow an almost-constant-velocity model independently in X and Y as described in Section 2.2.2. As there is only a single target, the state of the target at time t is denoted by x_t. The measurements were collected at a constant sampling rate and the state evolution process was defined by

$$\psi(x_t|x_{t-1}) = \mathcal{N}\left(x_t; Fx_t, Q\right), \tag{3.35}$$

where $\mathcal{N}(t; \mu, \Sigma)$ is a multivariate Gaussian,

$$F = \begin{bmatrix} F_{2t} & 0 \\ 0 & F_{2t} \end{bmatrix} \quad \text{with} \quad F_{2t} = \begin{bmatrix} 1 & \tau_t - \tau_{t-1} \\ 0 & 1 \end{bmatrix},$$

and

$$Q = q \begin{bmatrix} Q_{2t} & 0 \\ 0 & Q_{2t} \end{bmatrix} \quad \text{with} \quad Q_{2t} = \begin{bmatrix} \frac{(\tau_t - \tau_{t-1})^3}{3} & \frac{(\tau_t - \tau_{t-1})^2}{2} \\ \frac{(\tau_t - \tau_{t-1})^2}{2} & \tau_t - \tau_{t-1} \end{bmatrix}.$$

The positional measurement function of the sensor was linear and Gaussian,

$$\zeta(z_n|x_t) = \mathcal{N}\left(z_n; Hx_t, R\right), \tag{3.36}$$

Table 3.4: Monte Carlo Position RMS results

v	PMHT-t	KF1	KF2	PF 20 samples	PF 100 samples	PF 500 samples
0.2	3.64	20.68	8.87	4.83	3.82	3.68
0.4	3.32	8.09	7.78	4.21	3.67	3.58
0.6	2.97	4.87	6.64	3.91	3.52	3.46
0.8	2.64	3.80	5.20	3.52	3.33	3.32
1.0	2.21	3.13	3.13	3.13	3.13	3.13

with $H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ and $R = rI$.

The initial state of the target was $x_0 = [500, 15, 2000, -10]^{\mathsf{T}}$. The target estimates were initialised with the true initial position and with an initial state covariance of $P_0 = diag(100, 4, 100, 4)$. The process noise intensity was $q = 1/10$. The measurement noise variance was $r = 9$. The measurement sequence length was $N = 50$ and the true measurement times were $\tau_n = 2n, n = 1, ..., N$. The true parameters of the prior distributions for the timing error were $\mu = 2$ and $\lambda = 2$.

The performance of the algorithms was measured by three different time averaged RMS error metrics. The first is the time averaged RMS position error. The second is the time averaged RMS of the timing error mean (μ) and the third is the RMS of the timing error precision (λ). These RMS results were collected over 1000 realisations for all the algorithms described.

The variable v represents the probability of a correct measurement time being available. For example, in the case where $v = 1$, the correct measurement time was always available. Monte Carlo simulations were performed with different values of v to assess the tracking performance as the number of accurate timing measurements decreases.

The results for the timing error tracks are shown in Tables 3.4, 3.5 and 3.6 for the state, timing error mean and timing error precision, respectively.

Table 3.4 shows that the PMHT-t algorithm achieves overall better performance than the PF algorithm with 500 samples, which performed the best out of the alternate algorithms. The RMS error is marginally lower when v is low, but the RMS error is much lower as v is increased. When $v = 1$, the accurate time stamps are always available and the improvement in RMS position error was because the PMHT-t used a sliding window whereas the other algorithms were recursive.

Table 3.5: Monte Carlo Timing Error Mean RMS results

v	PMHT-t	KF1	KF2	PF 20 samples	PF 100 samples	PF 500 samples
0.2	0.31	0.54	0.54	0.61	0.36	0.30
0.4	0.20	0.40	0.40	0.35	0.27	0.26
0.6	0.17	0.33	0.33	0.31	0.26	0.26
0.8	0.16	0.29	0.29	0.27	0.26	0.26
1.0	0.16	0.25	0.25	0.25	0.25	0.25

Table 3.6: Monte Carlo Timing Error Precision RMS results

v	PMHT-t	KF1	KF2	PF 20 samples	PF 100 samples	PF 500 samples
0.2	1.00	2.85	2.85	1.69	1.58	1.47
0.4	1.01	2.07	2.07	1.62	1.45	1.41
0.6	1.02	1.65	1.65	1.39	1.38	1.34
0.8	0.99	1.46	1.46	1.35	1.32	1.31
1.0	0.92	1.29	1.29	1.29	1.29	1.29

Similarly, in Tables 3.5 and 3.6, the PMHT-t gave improvements in the estimation of the timing error mean and timing error precision compared with the PF algorithms. In the case where $v = 0.2$, the PMHT-t mean estimate has a very similar RMS error as the PF algorithm with 500 samples. For all other values of v, the PMHT-t has much improved performance over the alternate algorithms.

3.6 Conclusion

This chapter has introduced a method for using PMHT to perform target tracking in the situation where the temporal information is noisy or unreliable. The key idea is to use the PMHT to associate measurements to time instants as well as to targets by treating both assignments as missing data.

The PMHT-t association algorithm determines probabilities for each pairing of measurement and target at each possible time and then estimates the target states by taking the expectation over these assignments.

Simulation experiments were used to demonstrate the effectiveness of PMHT-t to track us-

ing measurements with time uncertainty. This method was demonstrated with simulations and improved performance was observed compared with the standard PMHT.

Further simulation experiments dealt with the scenario where true measurement times were available occasionally while noisy measurement time was available for all measurements. An estimate of the time difference between the true and noisy measurement times was represented by a set of time error parameters which were updated each time an accurate time measurement was available. The PMHT-t was used to process the measurements as a sliding window batch to simultaneously estimate the track position and velocity and the parameters of the measurement noise process and in particular the measurement time stamps.

The PMHT-t was shown to give good performance compared to recently proposed alternative algorithms such as Kalman filtering and the particle filter. The PMHT-t gave improvements compared with the particle filter, even when the number of particles was very high. This scenario has shown the flexibility of the PMHT-t to performing target tracking where the temporal information is noisy or unreliable.

Chapter 4

PMHT Path Planning

4.1 Introduction

Multiple platform path planning is a problem that arises in many applications including search and rescue, coordinated surveillance, multiple platform simultaneous localisation and mapping (SLAM) and resource dissemination (the travelling salesman problem). Independent of the application, the goal is to schedule multiple mobile resources with dynamic constraints to cover an area in an efficient manner.

There are many strategies to control a single moving platform to intelligently explore an environment. Most of these strategies take different factors into account before choosing an action, such as the localisation error of the platform's sensors or the information gain potential through the measurements that these sensors may collect. Generally these approaches enumerate a collection of hypotheses based on feasible motion of the platform and use a cost function as the decision criterion.

Active control, also known as robotic exploration, is the problem where the platform must choose its movement path while performing SLAM. In a predefined fully modelled environment, optimising the robot's path is well understood, but in practice, exploring robots have to cope with incomplete maps.

Active control schemes are designed with several tradeoffs to determine where to move. These tradeoffs include: the expected gain in map information; the expected gain in localisation accuracy; and the cost incurred in collecting this information and guiding the platform to the desired location (both in time and energy). The main goals are to increase localisation accuracy,

increase the map information and to choose the optimal (in some sense) path to travel.

An ideal scheme would meet all of the three goals above, but this is difficult because they conflict. For example, one major goal may be for the platform to map out an area as quickly as possible. To do this, the platform would need to be fast, optimise the amount of area covered by its sensors, and not revisit areas already covered. This compromises localisation accuracy because revisiting previously mapped landmarks is the key to accurate localisation.

4.1.1 Background

Different groups of researchers have tackled this problem by focusing on different aspects and combinations of these tradeoffs. Examples include classic exploration strategies that focus on moving the robot to build the map information as quickly as possible with localisation being less important, such as [Yam97]. An alternative based on the opposite strategy is Active Localisation [CKK96], where localisation is important and the platform's movement is scheduled in a way that minimises the expected localisation uncertainty. Other methods seek to optimise the view points of the robot, to maximise the expected information gain to build the map, and to minimise the uncertainty of the platform within the grid cells of the map [GKC03]. These strategies may be greedy one-step ahead or N-step ahead and are generally for single platforms.

One popular strategy focused on the active control paradigm is known as integrated exploration. This involves the platform performing both localisation and mapping simultaneously, whilst making local movement decisions in order to minimise the error in estimates of the location of the platform and the landmarks being mapped.

Integrated exploration for single platforms has been the focus of much work in the robotics community. Examples include EKF based approaches [FLS99] and using the information filter version of the Kalman filter [MWBDW02], [BS06a] and [SR05].

The common approach in the above is to introduce a cost function which balances the cost of reaching a selected position with either the possible loss of platform localisation accuracy or the potential information gain. These cost functions then produce a local decision on where the platform should move.

It has been widely acknowledged within the autonomous vehicle research community that the use of multiple cooperating vehicles for exploration tasks has many advantages over a single vehicle. Multiple vehicles have the potential to explore and map an environment more quickly than a single vehicle, are more robust to failures, and provide a broader field of view in dynamic

environments. To achieve cooperative exploration, the key problem is to choose appropriate actions for the platforms so that they simultaneously explore different regions of the environment optimally. Some of the single platform strategies have been extended to the multiple platform case, such as [BMF+00] which aims to build a map as quickly as possible, and [BS09] which uses information gain to coordinate the multiple platforms.

Multiple platform SLAM with active control has also been actively researched, [FNL02], [MR05], [TL05] and [BS06b]. These papers focus on accounting for the extra position information available when multiple platforms operate simultaneously. [FNL02] and [MR05] have also determined performance bounds for the collaboration of the platforms. [FNL02] determined a lower bound on platform position errors, which enables calculations of the number of platforms required to accomplish a certain task. In contrast, [MR05] determined the upper bounds for the position error of the platforms and proposed to utilise the calculation of the bound to determine the control of the platform to optimise the SLAM results.

[TL05] and [BS06b] have used the extended information filter (EIF) version of the Kalman filter. The EIF is a mathematically equivalent form of the EKF which uses an information matrix and information vector to represent the estimate rather than the standard mean state and covariance matrix. The information state and matrix are the first and second moments of the log likelihood. The advantage is that the EIF can optimally combine two estimates of a state together by simply adding their information matrices and adding their information vectors, provided that the errors in each estimate are not correlated with one another. This means that the information fusion of an arbitrary number of platforms becomes a simple sum of the estimates of each platform.

Another advantage of the EIF is that the information matrix is the second moment of the log likelihood, and so it can be related to entropy. The information gain from an observation is not dependent on the observation itself, but only the sensor accuracy. Therefore, it is possible to hypothesise the expected information if a particular platform action is adopted and hence control the platform to minimise entropy in the map.

The technique of using information gain to make decisions on platforms has been explored further using “partially observable Markov decision processes” (POMDP). Within the path planning context, POMDP is a control method based on observations of the platform states rather than complete knowledge of them. The POMDP system makes decisions based on a cost function combining the rewards and costs related to making a particular decision. POMDPs have been used in robotic navigation in [SK95] and [TM02].

In the above path planning techniques, the strategy has been to enumerate a collection of hypotheses based on what the platform's sensors are able to detect or the feasible motion of the platform and use a cost function as the decision criterion. These techniques provide a hard decision and generally a 1-step ahead solution to the path planning problem and lack constraints such as platform motion models.

4.1.2 Proposed Approach

This chapter introduces a new approach to multiple platform path planning. Given known initial conditions for each platform (position, speed, heading) and a set of discrete locales of interest whose position is known, the Probabilistic Multi-Hypothesis Tracker is used to design trajectories for the platforms to cooperatively visit the locales.

The novelty in the PMHT path planning approach is to treat the locales as measurements and the platforms as targets and then employ a multi-target tracking algorithm to perform data association, namely to associate the locales with platforms. This ensures that each of the locales is visited by at least one of the platforms, while constraining the motion of the platforms to a realistic dynamic model. The difficulty is that the locales have no intrinsic temporal relationships. Whereas in tracking the usual assumption is that the input information is noisy position estimates collected at known times, here there are no times associated with each locale; there is not even a preferred order in which to visit them. In order to overcome this problem, hypothesised time-stamps are treated as hidden variables and the PMHT is used to associate the locales to platforms and times simultaneously. These hypothesised time-stamps can represent the order of visitation of the various platforms to the locales.

The advantage of the PMHT over alternative data association techniques is that it has linear complexity in the number of targets, the number of measurements per frame, and in the number of frames. In the context of the planning problem, PMHT allows for batch data association over hundreds of time steps with multiple platforms in a field of hundreds of locales. It is also built on an EM framework which is amenable to the incorporation of the missing temporal information.

Sections 4.2 and 4.3 formulate the problem and Section 4.4 derives the PMHT path planning algorithm and provides some qualitative performance examples. Section 4.6 investigates parameters that can be varied to alter the performance of the planned trajectories. The scalability and performance of the algorithm is assessed in Section 4.7 as the number of locales increases.

In Section 4.8, the algorithm's performance is assessed quantitatively and compared with a sophisticated alternative method. The alternative approach considered is to treat the path planning as a Travelling Salesman Problem (TSP) and find the TSP solution via a Genetic Algorithm. The TSP is a common optimisation problem whereby an agent must visit a set of known locations exactly once and in the order that minimises the distance travelled. The multiple platform path planning problem can be treated as a TSP with multiple agents that must cooperatively visit each location in a manner that minimises the total collective distance travelled. No efficient algorithm exists to find an optimal solution to the TSP, therefore a randomised approximate optimisation method is used, namely a Genetic Algorithm (GA).

A sliding window batch version of the PMHT path planner is considered in Section 4.9. Section 4.10 introduces the use of a particle filter as the estimator for the PMHT path planner to explore indoor environments. Implementation strategies are demonstrated via several example scenarios.

Finally in Section 4.11, the PMHT path planner is extended from using a uniformly weighted collection of locales to a non-uniformly weighted set of discrete locales by introducing the concept of locale priority. This extension produces a PMHT path planner that no longer uses the notion of artificially generated locales to guide the platform trajectories, but instead driven by an intensity function over the area to be explored.

4.2 Problem Formulation

Assume there is a region to be explored by M multiple sensor platforms cooperatively in T time steps. A set of N locales of interest have been created at known locations throughout this region to be visited by these platforms. These locales may be an artificial construct that provide a means of guiding the sensors: visiting the locales is equivalent to exploring the region. This set of locales may be a grid of points distributed uniformly across the area, such as in Figure 4.1, or a more precisely placed set of points dependent on previous knowledge.

Let the state of platform m at time t be denoted x_t^m, the set of all states for platform m be denoted X^m, and the set of all states be denoted X. For this problem, the state consists of the position, velocity and acceleration in the X and Y domain, namely

$$x_t^m \equiv [x_t^m, \dot{x}_t^m, \ddot{x}_t^m, y_t^m, \dot{y}_t^m, \ddot{y}_t^m]^{\mathsf{T}} \tag{4.1}$$

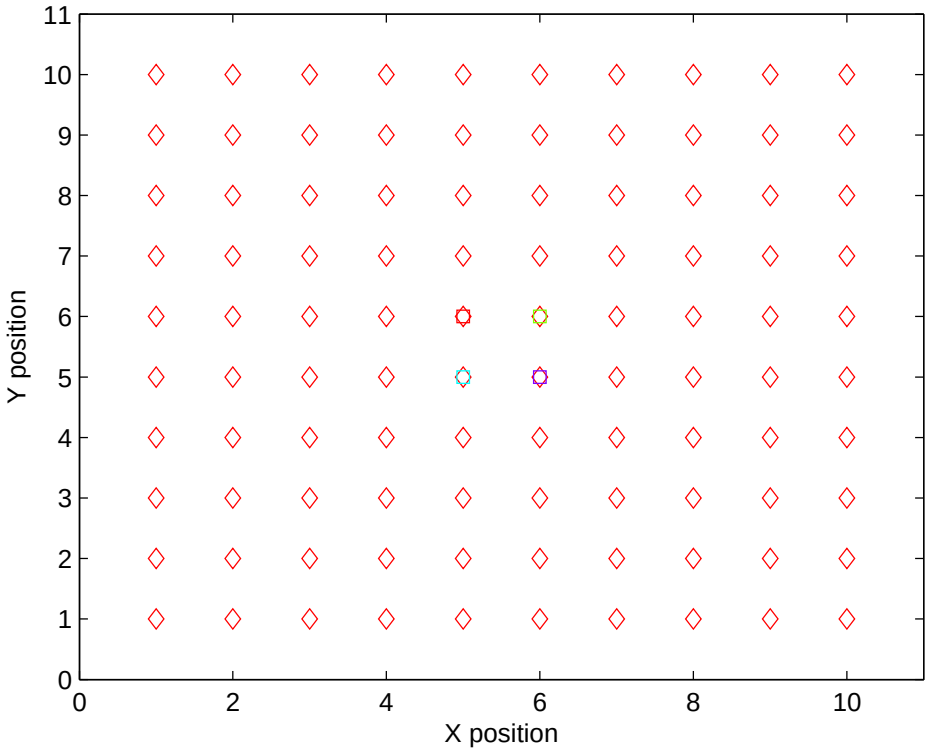


Figure 4.1: Exploration using locales of interest

Assume that the prior distribution of the state of each platform is known and is given by $\psi_0^m(x_0^m)$ for platform m. The platform dynamics are also assumed to be known and can be described by the evolution probability density function (pdf) $\psi_t^m(x_t^m|x_{t-1}^m)$.

Let the location of the nth locale, denoted by z_n, be given by

$$z_n = [z_n^x, z_n^y]^\mathsf{T}, \tag{4.2}$$

where z_n^x and z_n^y are the x and y positions of the locale respectively. It is assumed that the locales are known exactly. Let the platform assigned to locale n be $k_n \in 1 \dots M$ and the time assigned for platform k_n to visit locale n be $\tau_n \in 1 \dots T$. Let Z denote the set of all locales, K denote the set of all platform-to-locale assignments and τ denote the set of all time-to-locale assignments. The goal of path planning is to choose assignments of the platform and time for each locale in such a way that the trajectories required to achieve this visit sequence are in some sense optimal, or at least "good". This implies a joint optimisation problem over X, K and τ.

4.3 Path planning problem

As stated above, the planning problem is one of joint optimisation. Note that the assignments of platforms to locales and times are nuisance parameters: the aim is to plan the platform trajectories, X, but we are forced to consider a joint assignment problem to do this. If the assignments K and τ had been specified, then the platform states could be independently planned by fitting a smoothed trajectory through an implied set of waypoints. There would be no interaction between platforms in this planning because the indices effectively divide the locales into subsets for each platform.

As these assignments are unknown, the smoothed platform trajectories need to be estimated while simultaneously finding the best assignment values. To do this, a utility function is defined as $f(X, Z, K, \tau)$ and the goal is to define the planned paths as the set of parameters to maximise this function.

The utility function is composed of three cost function terms which are of importance to planning the paths for multiple platforms. The first involves the cost of controlling the platform dynamics, the second is the cost of each platform to visit each of the locales and finally the cost of the platform to locale assignment indices. Each of these terms are now described in more detail.

The first term describes the motion of the platforms. This term is defined as

$$\sum_{m=1}^{M}\sum_{t=1}^{T} g^x(x_t^m, x_{t-1}^m) = -\sum_{m=1}^{M}\sum_{t=1}^{T} C^x \left\| x_t^m - x_{t-1}^m \right\| \tag{4.3}$$

where the function on the right results in g^x being high when the distance between x_t^m and x_{t-1}^m is small. Note that C^x is a constant which scales the importance of this cost term. With the summations, this first term quantifies the total cost of the platforms motion, given their initial states.

The second term of the utility function describes which platforms visit each of the locales. A locale is considered "visited" if one of the platforms passes sufficiently close to the locale. This term is therefore dependent on the location of platform m at time t (as assigned by k_n and τ_n respectively) and the position of locale n to be visited. This term is defined as

$$\sum_{n=1}^{N} g^z(x_{\tau_n}^{k_n}, z_n) = -\sum_{n=1}^{N} C^z \left\| x_{\tau_n}^{k_n} - z_n \right\|, \tag{4.4}$$

where the function on the right results in g^z being high when the distance between $x_{\tau_n}^{k_n}$ and z_n is small. Note the relative scaling factor C^z which scales the importance of this cost term. This term can be used to control how closely the platforms approach each discrete locale position.

The final term of the utility function allows for a penalty based on the assignment indices themselves. For example, it is undesirable that all of the locales be assigned to a single platform or that all of the locales be visited at the same time. At this stage, the decision over the form of this term is deferred and simply denoted as

$$g^{(k,\tau)}(K, \tau). \tag{4.5}$$

Combining the three terms, the overall utility function is

$$f(X, Z, K, \tau) = \sum_{m=1}^{M}\sum_{t=1}^{T} g^x(x_t^m, x_{t-1}^m) + \sum_{n=1}^{N} g^z(x_{\tau_n}^{k_n}, z_n) + g^{(k,\tau)}(K, \tau) \tag{4.6}$$

This utility function in equation (4.6) is similar to the log-likelihood in a tracking problem. In tracking, the aim is to estimate the states of targets $x_{1:T}^{1:M}$, which evolve according to a Markov chain, based on measurements $z_{1:N}$ with ambiguous origin. The true source of measurement n

is k_n and in the time ambiguous context, the true time is τ_n. By taking the log of the complete data likelihood in equation 3.2 from Chapter 3, the tracking log-likelihood is

$$\mathcal{L} = \sum_{m=1}^{M}\sum_{t=1}^{T}\log\{p(x_t^m|x_{t-1}^m)\} + \sum_{n=1}^{N}\log\{P(z_n|x_{\tau_n}^{k_n})\} + \log\{P(\tau_{1:N}, k_{1:N})\} \tag{4.7}$$

In the planning case, the utility function f is to be maximised over variables x, k and τ with fixed parameters z. In the tracking case, $\mathcal{L}$ is maximised over random variables x, k and τ for a particular realisation of the random variables z. If the mathematical form of $g^x(x_t^m, x_{t-1}^m)$ is the same as $\log\{p(x_t^m|x_{t-1}^m)\}$, $g^z(x_{\tau_n}^{k_n}, z_n)$ as that of $\log\{P(z_n|x_{\tau_n}^{k_n})\}$ and finally $g^{(k,\tau)}(K, \tau)$ is the same as $\log\{P(\tau_{1:N}, k_{1:N})\}$, then the solution to maximising f can be obtained by maximising $\mathcal{L}$ and substituting the locales for measurements.

Notice that the measurements in the tracking problem are random variables whereas the locales in the planning problem are user defined fixed parameters. This may make the reader nervous. However, it has not been implicitly assumed that a deterministic parameter to be random. Instead, the tracker has been treated as a numerical tool that can be used to optimise a function of a certain mathematical form. The assumptions used to create the tool do not change the fact that it optimises the same equation as encountered in the path planning problem. Nevertheless, care should be taken. For example, the Kalman Smoother will report a covariance matrix for the target state. In the tracking context this matrix represents the second order moment of the estimation error. In the path planning context it is an intermediate gain parameter with no intrinsic meaning at all.

A version of PMHT-t is now developed in the complete absence of time information. This algorithm will be the numerical tool used to optimise the path planning utility function and will be referred to as PMHT-pp.

4.4 PMHT for multiple platform path planning

Consider a multi-target tracking problem where there is no clutter and where the measurements are received with no temporal information at all. As stated above, the solution to this problem will provide a numerical tool to solve path planning. This problem can be considered to be a special case of the uncertain time-information problem addressed in chapter 3. Whereas the tracker in chapter 3 (i.e. PMHT-t) had access to noisy time information, here the problem is

equivalent to one where the time information is so noisy that it is independent of the true measurement time, i.e. there is no time information at all. Based on this intuition, it is observed that the target model and the assignment model are the same as for PMHT-t, but the measurement model is different because there is no z_n^τ. For completeness we briefly derive the algorithm for this modified problem.

As before, the complete data are (X, τ, K, Z), the incomplete data are (X, Z) and (τ, K) are the missing data. The auxiliary function is the expectation of the complete data log-likelihood over the missing data, which now takes the form:

$$Q(X, \Pi^\tau, \Pi^k | \hat{X}(i), \hat{\Pi}^\tau(i), \hat{\Pi}^k(i)) = \sum_K \sum_\tau P(\tau, K | \hat{X}(i), Z; \hat{\Pi}^\tau(i), \hat{\Pi}^k(i)) \log P(X, \tau, K, Z; \Pi^\tau, \Pi^k), \quad (4.8)$$

where the summation is over all permutations of the assignment variables τ and K. Recall that $\hat{X}(i)$ denotes an estimated variable on the ith EM iteration and similarly for $\hat{\Pi}^\tau(i)$ and $\hat{\Pi}^k(i)$.

Due to the independence assumptions, the complete data likelihood becomes:

$$P(X, \tau, K, Z; \Pi^\tau, \Pi^k) = P(X) P(\tau; \Pi^\tau) P(K; \Pi^k) P(Z | X, \tau, K), \quad (4.9)$$

where

$$P(X) = \prod_{m=1}^{M} \left\{ \psi_0^m(x_0^m) \prod_{t=1}^{T} \psi_t^m(x_t^m | x_{t-1}^m) \right\}, \quad (4.10)$$

$$P(\tau; \Pi^\tau) = \prod_n \pi^\tau_{n\tau_n}, \quad (4.11)$$

$$P(K; \Pi^k) = \prod_n \pi^k_{nk_n}, \quad (4.12)$$

$$P(Z | X, \tau, K) = \prod_{t=1}^{T} \prod_{n=1}^{N} \zeta(z_n | x_{\tau_n}^{k_n}) \quad (4.13)$$

Note again that this is the same as for the PMHT-t in chapter 3 equations (3.2)-(3.6) except that the measurement likelihood (4.13) only contains a spatial measurement term whereas the corresponding (3.6) also contains a temporal measurement term.

The conditional probability of the missing data, $P(\tau, K | \hat{X}(i), Z; \hat{\Pi}^\tau(i), \hat{\Pi}^k(i))$, can be de-

termined using Bayes' Rule:

$$
\begin{aligned}
& P(\tau, K|\hat{X}, Z; \hat{\Pi}^\tau, \hat{\Pi}^k) \\
&= \frac{P(\hat{X}, \tau, K, Z; \hat{\Pi}^\tau, \hat{\Pi}^k)}{\sum_{\tau,K} P(\hat{X}, \tau, K, Z; \hat{\Pi}^\tau, \hat{\Pi}^k)} \\
&= \frac{P(\hat{X})P(\tau; \Pi^\tau)P(K; \Pi^k)P(Z|\hat{X}, \tau, K)}{P(\hat{X})\sum_{\tau,K} P(\tau; \Pi^\tau)P(K; \Pi^k)P(Z|\hat{X}, \tau, K)} \\
&= \frac{P(\tau; \Pi^\tau)P(K; \Pi^k)P(Z|\hat{X}, \tau, K)}{\sum_{\tau,K} P(\tau; \Pi^\tau)P(K; \Pi^k)P(Z|\hat{X}, \tau, K)} \\
&= \frac{\prod_{n=1}^{N} \pi^\tau_{n\tau_n}\pi^k_{nk_n}\zeta(z_n|x^{k_n}_{\tau_n})}{\sum_{\tau,K}\prod_{n=1}^{N} \pi^\tau_{nt}\pi^k_{nm}\zeta(z_n|x^m_t)} \\
&= \prod_{n=1}^{N} \frac{\pi^\tau_{n\tau_n}\pi^k_{nk_n}\zeta(z_n|x^{k_n}_{\tau_n})}{\sum_{r=1}^{T}\sum_{s=1}^{M}\pi^\tau_{nr}\pi^k_{ns}\zeta(z_n|x^s_r)} \equiv \prod_{n=1}^{N} w_{n\tau_n k_n}
\end{aligned}
\tag{4.14}
$$

where the iteration index (i) is suppressed for clarity.

Thus the conditional probability of the assignments is given by the product of individual per measurement *weights*. Each weight, w_{ntm}, is the normalised likelihood of the nth measurement from platform m at time t and following the discussion of Section 4.2, since the nth measurement is fixed and in the position of the nth locale, w_{ntm} may be interpreted as the probability that the mth platform visits the nth locale at time t. The numerator of the weight is the product of the assignment priors and the positional measurement likelihood. The difference between this weight function and that of the PMHT-t in (3.7) is that the locales do not have a time-stamp, so there is no pmf in terms of the time. Equation (4.14) corresponds to the PMHT-t weights (3.7) for an uninformative time measurement, $\zeta^\tau(z^\tau_n|\tau_n) = \frac{1}{T}$.

Substituting (4.9) and (4.14) into (4.8) leads to the auxiliary function to be maximised:

$$\begin{aligned} & Q(X, \Pi^\tau, \Pi^k | X(i), \Pi^\tau(i), \Pi^k(i)) \\ &= \log P(X) + \sum_{n,t,m} w_{ntm} \log \pi^\tau_{nt} + \sum_{n,t,m} w_{ntm} \log \pi^k_{nm} \\ &\quad + \sum_{n,t,m} w_{ntm} \log \zeta(z_n | x^m_t) \\ &\equiv Q_X + Q^\tau_\Pi + Q^k_\Pi \end{aligned} \tag{4.15}$$

where the choice of the Q_X term was discussed in Section 4.3. This auxiliary function is the same as the function in (3.8) except that there is no Q_τ term because there is no time information.

The term Q^τ_Π in (4.15) is given by

$$Q^\tau_\Pi \equiv \sum_{n,t,m} w_{ntm} \log \pi^\tau_{nt},$$

which is maximised subject to the constraints that $\sum_t \pi^\tau_{nt} = 1$ using a Lagrangian, resulting in the updated prior estimate

$$\hat{\pi}^\tau_{nt}(i+1) = \sum_{m=1}^{M} w_{ntm}, \tag{4.16}$$

i.e. the weights' relative frequency for time t.

Similarly, the Q^k_Π term results in a relative frequency estimate for the locale to platform assignment prior

$$\hat{\pi}^k_{nm}(i+1) = \sum_{t=1}^{T} w_{ntm}. \tag{4.17}$$

The remaining term, Q_X, couples the platform states and the locale positions and is given by

$$Q_X \equiv \log P(X) + \sum_{n,t,m} w_{ntm} \log \zeta(z_n | x^m_t). \tag{4.18}$$

For a Gaussian penalty function, $\zeta(z_n | x^m_t)$ with covariance R, it can be shown that this function is equivalent to the log likelihood of a tracking problem with known data association

Algorithm 4 PMHT-pp algorithm

1: Initialise platform state estimates, locale-to-platform assignment priors and locale-to-time assignment priors.
2: Calculate the assignment weight for each locale and platform at each scan, according to (4.14).
3: Update the assignment priors using (4.16) and (4.17).
4: Determine synthetic measurements and covariances for each platform at each scan using (4.20) and (4.21).
5: Update the platform state estimates using a Kalman smoothing algorithm over the synthetic measurements and covariances.
6: Repeat steps 2 to 5 until convergence of the auxiliary function (4.15).

[Dav07],

$$Q_X \equiv \log P(X) + \sum_{t,m} \log \tilde{\zeta}_t(\tilde{z}_t^m | x_t^m), \tag{4.19}$$

where the synthetic measurement, $\tilde{z}_t^m$, is given by

$$\tilde{z}_t^m = \frac{1}{\sum_{n=1}^{N} w_{ntm}} \sum_{n=1}^{N} w_{ntm} z_n, \tag{4.20}$$

and the synthetic measurement function, $\tilde{\zeta}(\cdot)$, has the same form as the measurement function with a scaled covariance given by

$$\tilde{R}_t^m = \frac{1}{\sum_{n=1}^{N} w_{ntm}} R. \tag{4.21}$$

The path for platform m is now refined by smoothing the synthetic measurements and covariances.

The planning algorithm consists of iteratively calculating assignment weights, w_{ntm}, and estimating the platform paths and assignment priors until convergence. The procedure is shown in Algorithm 4.

4.5 Simulation Results

The use of PMHT for multiple platform trajectory planning is now illustrated through simulation. In these simulations, the area to be explored is a 10 unit square box. A 10×10 uniform

grid of locales was defined and the PMHT algorithm described in the previous section was used to plan paths. Various combinations of initial platform position and number of platforms were considered and a selection are presented here. A listing of the Matlab source code to produce these results are in Appendix A.2.

The platform states were assumed to follow a constant acceleration model independently in X and Y as defined in Section 2.2.2, with $Q = 10^{-8}I$. The fitting penalty function was linear and Gaussian,

$$\zeta(z_n|x_t^m) = \mathcal{N}\left(z_n; Hx_t^m, R\right), \tag{4.22}$$

with $H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$ and $R = 10^{-2}I$. This corresponds to $C^z = -50$.

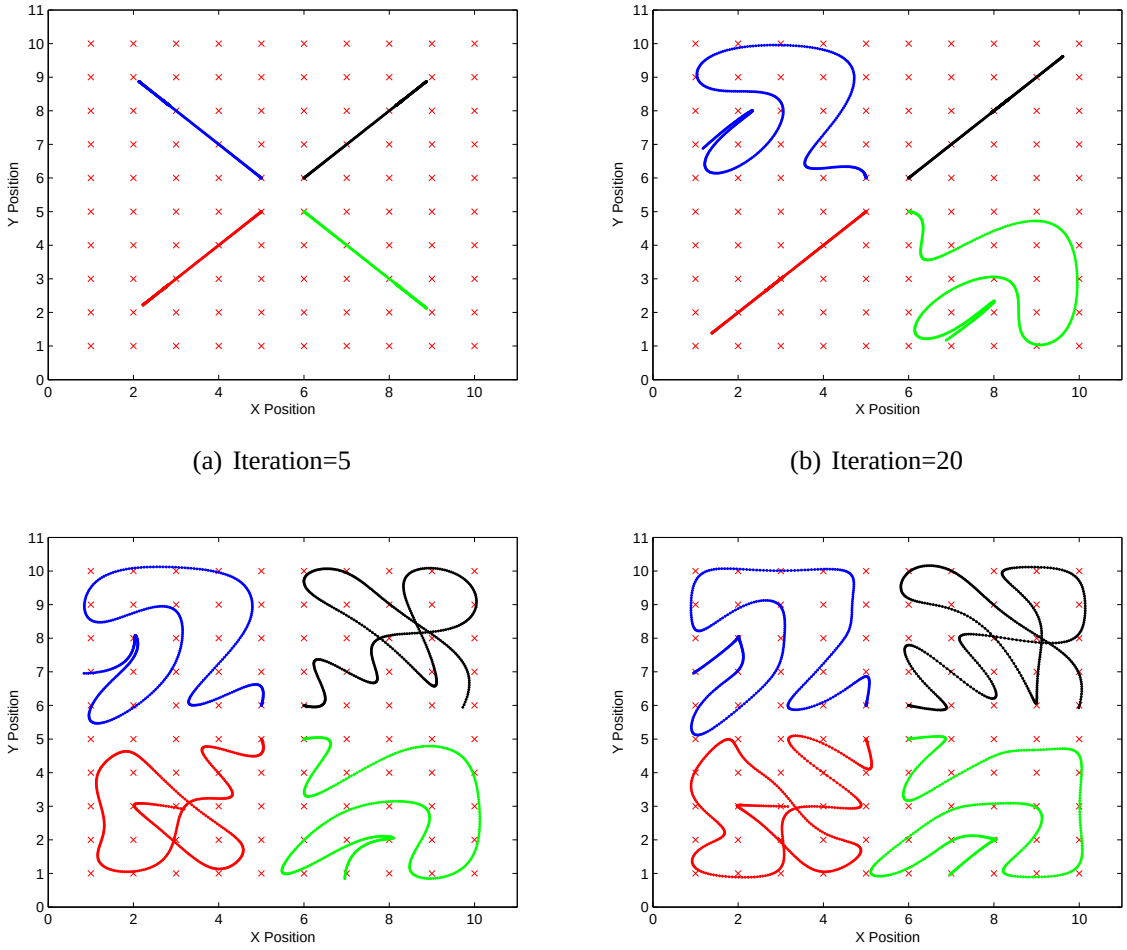
$T = 500$ time points were used. In each case, the platforms were initialised at a fixed position with zero speed and zero acceleration.

Figure 4.2 shows the evolution of the trajectories for four platforms with the EM iterations. The platforms were initialised in the centre of the scene. Initially, the assignment implicitly carves the area into quadrants and each platform moves in a straight line due to the symmetry. However, after a period, the paths diverge from the straight and narrow and eventually span each quadrant. The constraints on the platform dynamics inherent in the process model result in smoothly curving paths, as shown in the final solution.

Not all of the locales are closely visited by the platforms. There is a tradeoff between path smoothness and proximity to assigned locales and this is governed by the ratio of Q to R.

Figure 4.3 illustrates the iterated final solution for several other example scenarios. Figure 4.3(a) shows a similar scenario except this time the platforms begin outside the scene at the corners. Again due to symmetry, the map is divided into regular quadrants. Although the scenario is symmetrical, the planned trajectories do not converge to symmetrical paths. This is because each quadrant is an identical subproblem, with a set of locales and a platform initialised at the same position, but rotated. The optimisation of the platform to locale association is a chaotic system, where a slight perturbation may lead to a particular converged solution. The initial departure from straight paths in the early iterations is driven by finite precision rounding which introduces an element of variation into the process. This assertion was verified by rounding the weights in (4.14) to a smaller significant number which resulted in symmetrical but unconverged paths.

Figure 4.3(b) shows a three platform scenario where the platforms all begin on the left side, outside the map. In this case there is approximate symmetry for the outer platforms, which are



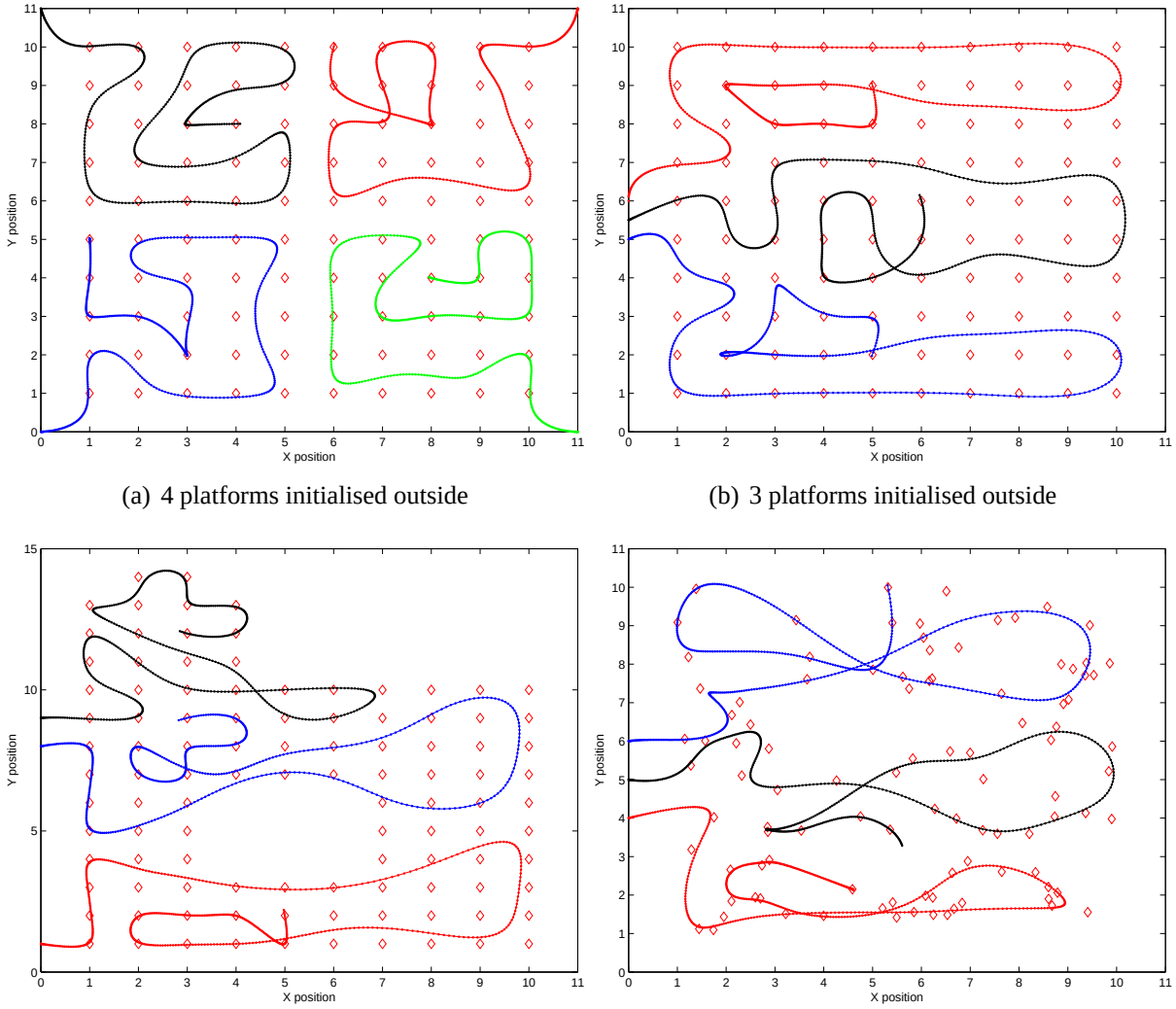
(a) Iteration=5

(b) Iteration=20

(c) Iteration=40

(d) converged

Figure 4.2: PMHT-pp assigned trajectories evolution for 4 platforms



(a) 4 platforms initialised outside

(b) 3 platforms initialised outside

(c) 3 platforms, irregular grid

(d) 3 platforms, random locales

Figure 4.3: Example planned trajectories

assigned almost mirror paths, but the path of the central platform produces some asymmetry.

Figure 4.3(c) shows a more complicated scenario without the symmetry of the previous examples. Here there is a hole in the middle of the region where no locales have been placed, and an extra group of locales in the North-West corner. Three platforms are initialised outside the region. Although there is no explicit constraint, the platforms avoid the empty region in the centre of the scene. However, the locale coverage on the far side of the empty region is more sparse as the platforms attempt to cover the area with a single loop. One of the platforms explores the North-West group of locales and the North-West corner of the square grid

Figure 4.3(d) shows the three platforms planning paths through a field of randomly distributed locales. The space is roughly divided amongst the three in the same manner as the regular grid, but the paths are chosen to pass through areas of high locale density and avoid areas where locales are sparse.

It is clear from the various examples that the paths of different platforms do not cross. This is due to a well known property of EM mixture fitting which essentially biases the platform paths away from each other [MK97]. This may be an undesirable feature of the planned paths. For example, in the context of SLAM, the division of the map into sub-maps allows for efficient coverage of the area without redundant observations, but the resulting picture needs to be stitched together and there may be significant errors in this process if there is little overlap between regions. There are several potential methods to address this deficiency, and this will be the subject of future investigation.

Figures 4.4 and 4.5 illustrate the estimated assignment priors in time and between platforms for the 4 platform case in Figure 4.3(a). Each locale has been strongly assigned to one of the platforms, i.e. the values of Π^k in Figure 4.4 are all very close to zero or one. This is consistent with the fact that none of the locales were visited by more than one platform.

In contrast, the temporal priors in Figure 4.5 are more spread. This implies that there were several time points in a row where a platform was close enough to be associated with the locale. In some instances, particularly towards the end of the time sequence, the prior was smeared across tens of time steps. This indicates that the platform has slowed down and takes a long time to pass by a locale. This occurs when there are many available time steps remaining, but the planner has already visited most locales. Also note that some locales are visited more than once, which leads to two bright patches on a single row of the image. This happens when a single platform crosses its own path.

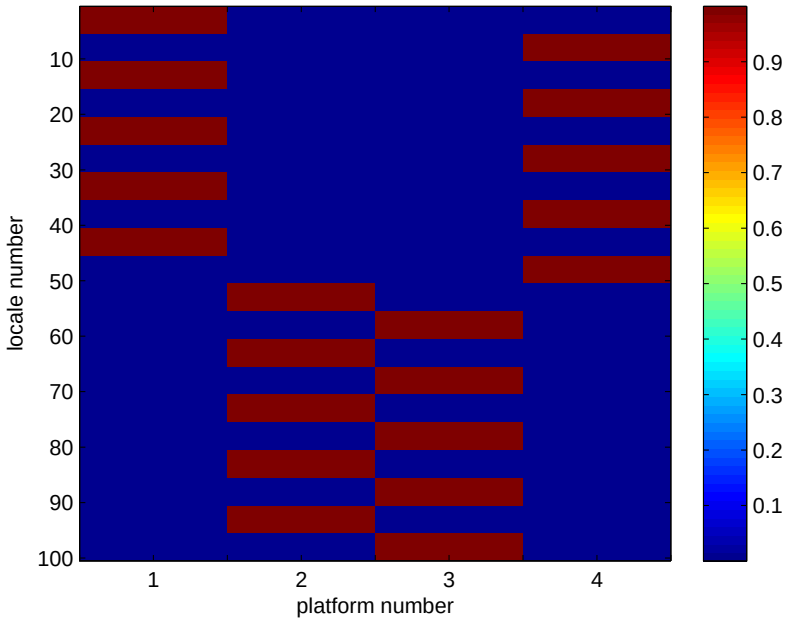


Figure 4.4: Converged estimates of Π^k from the simulation in Figure 4.3(a)

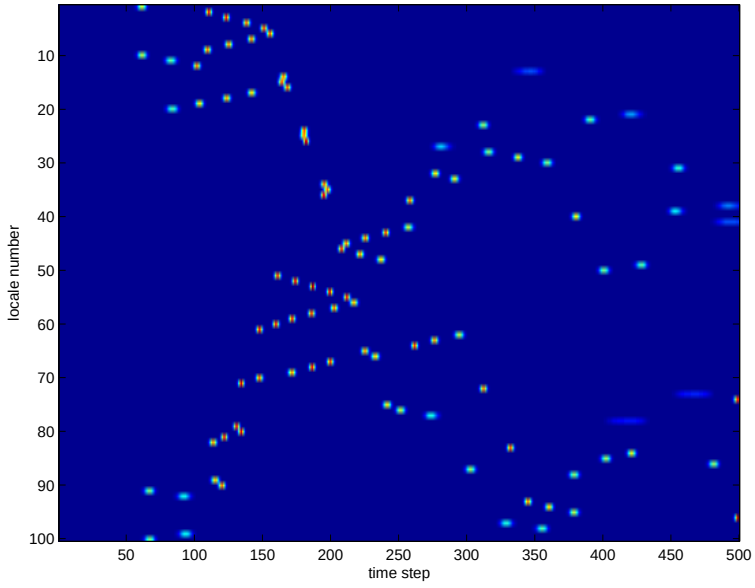


Figure 4.5: Converged estimates of Π^τ from the simulation in Figure 4.3(a)

4.6 Tradeoff between smoothness and proximity

An observation of the simulation results in Section 4.5 shows that not all of the locales are closely visited by the platforms. The platform trajectories produced by the PMHT-pp algorithm are governed by the system and measurement equations used to model the motion of the platforms.

The probabilistic system model is

$$p(x_t|x_{t-1}) = \mathcal{N}(x_t; Fx_{t-1}, Q_{t-1}) \tag{4.23}$$

with state transition matrices F and system covariance Q_t.

The fitting penalty function which is analogous to the probabilistic measurement model takes the form

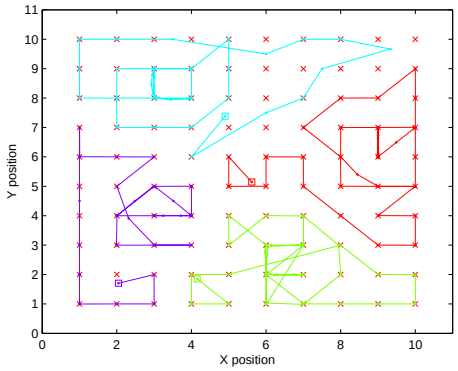
$$g^z(x_t^m, z_n) = log\{\mathcal{N}(z_n; Hx_t^m, R)\} \tag{4.24}$$

with measurement function H and measurement covariance R.

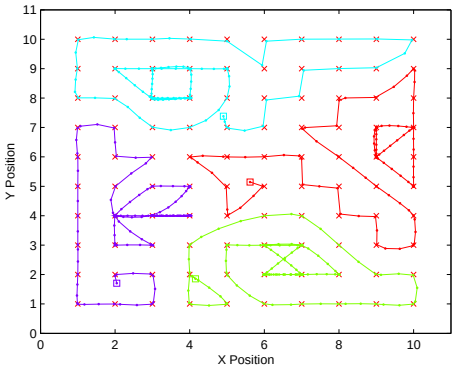
There is a tradeoff between path smoothness and proximity to assigned locales and this is governed by the ratio of Q to R. A higher value of Q allows the platform trajectory to manoeuver more sharply, leading to tighter corners and thus allowing the path to pass closer to a locale. A higher value of R allows locales to be assigned to a platform that is farther away. Intuitively, we may consider the platform to have a field of view determined by R and large R values correspond to a broad field of view. When R is large the paths may not visit the locales as closely as when R is small.

Figure 4.6 shows the results of varying the ratio between Q and R on a scenario with four platforms and a grid of a hundred locales. When the ratio of $Q : R$ is high, the platforms have a tendency to travel near the position of the locales. As the ratio is decreased, the trajectories becomes smoother and the locales are considered visited despite being a greater distance from the trajectory. The total distance travelled is also shorter due to this tradeoff as the platform does not have to travel through the position of the locale.

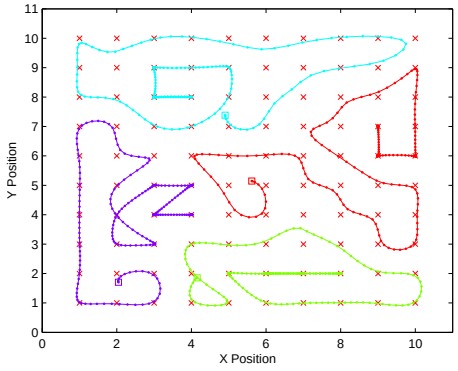
Figure 4.6(c) and 4.6(d) show the results from using the same $Q : R$ ratio, but different magnitudes. This identical result occurs because scaling R and Q by the same amount results in a scaling of the auxiliary function in (4.15). Scaling does not change the location of the maximum of the auxiliary function, merely the value at the maximum, so the state estimates are the same. For a given association, the paths are dependent only on the ratio $Q : R$. However the



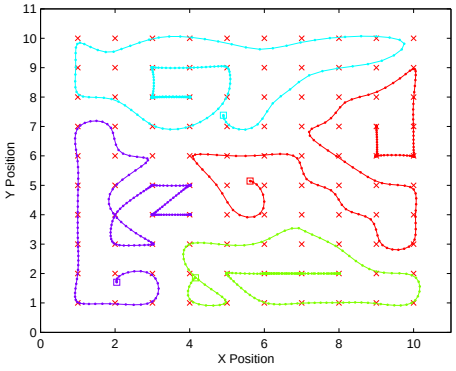
(a) $Q = 1000, R = 1$



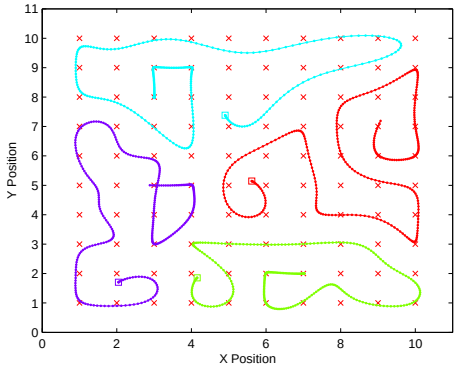
(b) $Q = 10, R = 1$



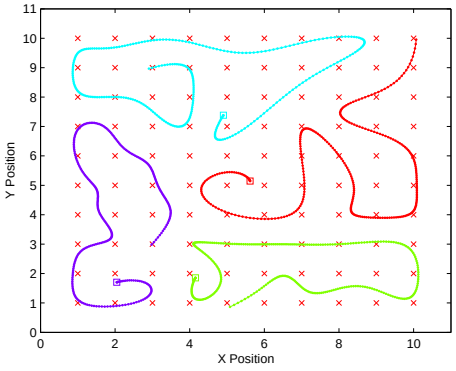
(c) $Q = 1, R = 10$



(d) $Q = 10, R = 100$



(e) $Q = 1, R = 1000$



(f) $Q = 1, R = 10000$

Figure 4.6: PMHT-pp Tradeoff using ratio between Q and R

magnitude of R may affect the associations between locales and platforms which would affect the final solution.

4.7 Locale Density Dependence

The locales are an arbitrary sampling of the exploration space and there is no theoretical basis for choosing a particular set of locale placements. In this section the performance of the PMHT-pp algorithm is investigated as the locale sampling is varied.

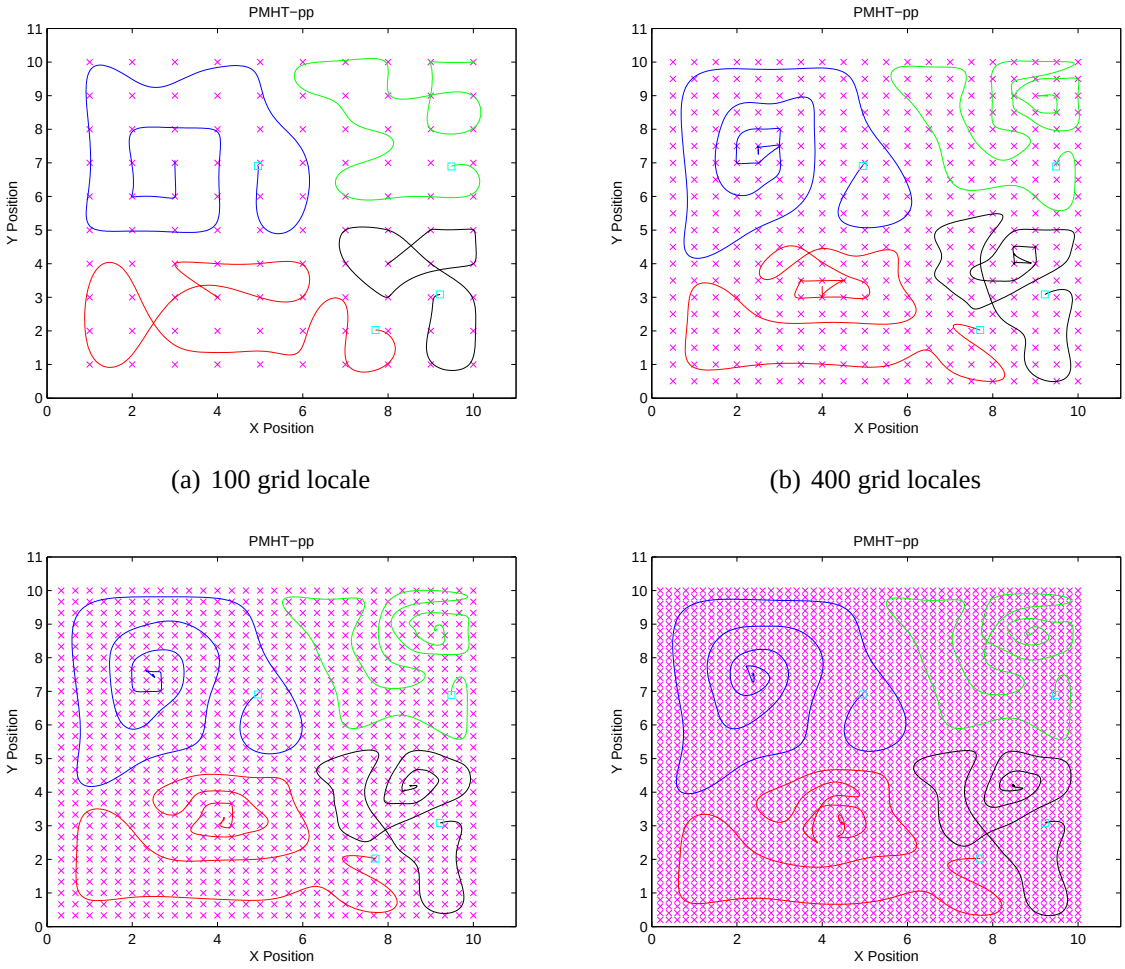
Of interest is the subjective quality of the paths, and objective performance metrics: the CPU time, and total distance required to travel for all the platforms.

As mentioned, the advantage of the PMHT over alternative data association techniques is that it has linear complexity in the number of targets, the number of measurements per frame, and in the number of frames. Due to this, the CPU time is expect to grow linearly as the number of landmarks is increased.

Two scenarios are considered. In the first, a uniform grid of locales was defined over a 10x10 area in the plane. Four platforms were initialised at random positions within the area and simulated identically as in Section 4.5.

Figure 4.7 shows the performance when 100, 400, 900 and 2500 locales are used in a grid. The position of the platforms in these four scenarios were initialised with their same individual locations and with zero velocity. The values of Q and R were $10^{-8}I$ and $10^{-2}I$ respectively. It can be observed that the paths do not cross and that the area is divided between the four platforms, despite the number of locales within the area. Another interesting observation is that as the number of locales is increased in the area, the paths still follow a similar path to cover the locales. As the system and measurement models remain the same, the coverage area of the platform remains the same, so despite the higher concentration of locales in a particular area, they may all be considered covered by the platform, hence the path still remains the same.

Monte Carlo simulations of the above scenario were performed, and the total distance travelled by the platforms averaged over 100 trials. These average total distances and the CPU time incurred for each algorithm are given in table 4.1. The CPU ratio comparing the CPU time to the number of locales is also given. As expected, as the number of locales increases, the amount of time to process the paths increases approximately linearly. As described above, the total distance travelled does not significantly increase with the locale density because the fitting functions has not changed. When the area is sparsely covered, the paths are not as complex so



(a) 100 grid locale

(b) 400 grid locales

(c) 900 grid locales

(d) 2500 grid locales

Figure 4.7: Trajectories with 4 randomly initialised platforms with varied grid of locales

Table 4.1: Monte Carlo results for a grid of locales

Number of Locales	CPU time	Total Distance	CPU Ratio
100	23	95	1
400	140	133	6
900	241	136	10
2500	575	140	25

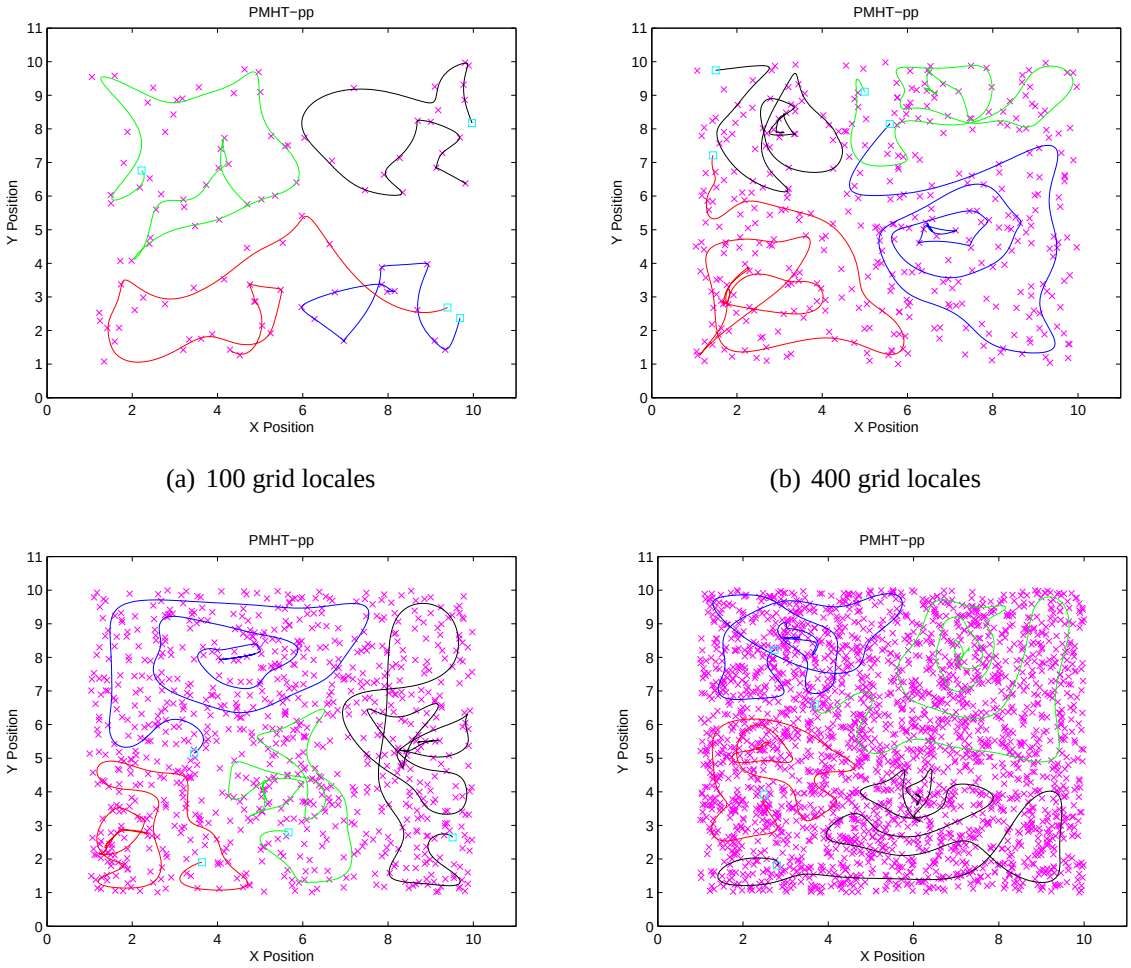
Table 4.2: Monte Carlo results for random locales

Number of Locales	CPU time	Total Distance	CPU Ratio
100	23	79	1
400	143	114	6
900	235	119	10
2500	573	125	25

there is less distance travelled in these cases.

In the second scenario, a set of locales were randomly positioned in a 10x10 area in the plane. Similarly, four platforms were initialised at random locations within the plane. Figure 4.8 shows the trajectories with a random set of locales. It can be seen that the paths are planned in a way that covers all the locales. Although the platforms do not travel to the exact position of each locale, it may still be considered to be within the sensor coverage area of the platform. Once again, the platforms tend to divide the area into quadrants and only cross if their initial position happens to be within the area of another platforms quadrant to be explored. Unlike the previous scenario, the paths do not look similar as the number of locales within the area is increased. This is because the locale positions are generated randomly and independently.

Table 4.2 shows the Monte Carlo simulation results of the random locale scenario, averaged over 100 trials. Similar to the grid scenario, as the number of locales increases, the amount of time to process the paths increases approximately linearly. Due to the random positions of the locales, the distance travelled by each of the individual platforms can vary. The total distance travelled followed a similar behaviour to the grid scenario.



(a) 100 grid locales

(b) 400 grid locales

(c) 900 grid locales

(d) 2500 grid locales

Figure 4.8: Trajectories with 4 randomly initialised platforms with varied number of random locales

4.8 Genetic Algorithm Solution to the Travelling Salesmen Problem

The purpose of this section is to compare the performance of PMHT path planning with a competent alternative. Because the planning algorithm is a multiple platform batch method, it is appropriate to compare it with a multiple platform batch alternative. One such approach is to treat each of the locales as a city in a Travelling-Salesman-Problem (TSP). In the TSP, a salesman must make a complete tour of a given set of cities in the order that minimises the total distance travelled. Each city must be visited exactly once.

The path planning problem in this chapter can be posed as a multi-TSP where multiple travelling salesmen cooperate to complete a tour of the set of cities. As before, each city is visited exactly once and the optimal solution is the set of paths that minimise the combined distance travelled by all salesmen (platforms). It is not required that the platforms return to their starting locations. It is important to highlight that the multi-TSP minimises the distance travelled without any dynamic constraints, this means that there is no smoothness imposed on the solution.

The TSP is well studied in optimisation and belongs to a class of problems where an optimal solution is expected to have exponential complexity in the number of cities. This section uses a Genetic Algorithm (GA) as an approximate optimisation method to efficiently seek a very good solution, rather than finding the optimal solution. The GA is a randomised search algorithm which is based on the mechanics of natural selection and natural genetics [Gol89]. The GA solution to the multi-TSP is summarised in Algorithm 5

The GA method is a randomised search, so it is not guaranteed to find the global solution in a finite number of iterations. However, multiple simulations have shown that the converged solution approaches a solution with similar total distance travelled. The converged solution may also vary if the GA is applied to the same problem multiple times because of its random nature. In the multiple path planning case, we anticipate that there may be numerous path-sets that have very similar length and it is not critical that the planning algorithm find the best of these.

The GA-TSP algorithm used in this paper was adapted from [Kir07]. This approach produces an ordered list of locales assigned to each of the platforms. An example result of the GA-TSP algorithm with four platforms in a 10x10 grid of locales is shown in Figure 4.9. Unlike the PMHT-pp algorithm, the assigned paths results have cross overs.

Algorithm 5 GA algorithm

1: Create an initial population. In this case, this amounts to a set of random permutations of potential paths for each of the platforms.
2: Calculate the fitness function for each of the population member. In this case, the fitness is the total distance travelled by all the platforms.
3: According to the fitness value, select the best population members and use them to form the next generation. Individual candidate solutions may be randomly modified (mutation) or pairs of candidate solutions may be mixed together (crossover). These operations include a combination of:

- Reverse the order of the locales,
- Swap segments of locales,
- Slide the order of locales,
- Randomly reassign locales to platforms.

4: Repeat steps 2 and 3 until the total population fitness converges.

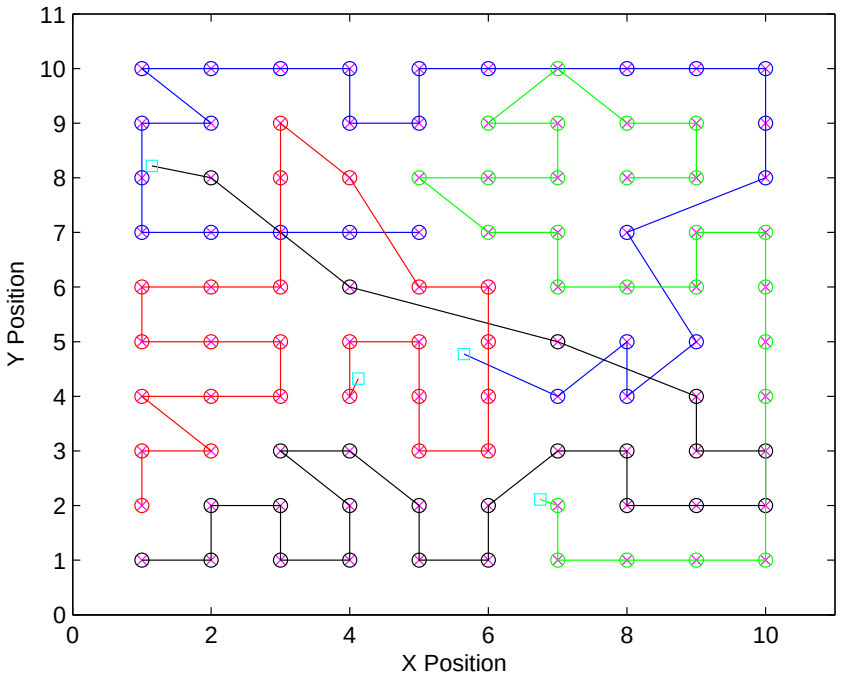


Figure 4.9: An example of GA-TSP with four platforms in a grid of locales

4.8.1 Genetic Algorithm TSP with PMHT Smoother

As mentioned above, the TSP formulation does not account for platform dynamics. It also demands that the platforms pass through each of the locales. In contrast, the PMHT-based algorithm plans paths that are sufficiently close to the locales based on a trade-off between path smoothness and the fitting penalty function.

In order to provide a balanced comparison, some modification to the GA-TSP is required. This was done by using the GA-TSP solution as initialisation data for the PMHT algorithm. The PMHT path planning algorithm was then run to produce smooth platform trajectories. By appropriate adjustment of the priors, as discussed below, the PMHT output can be guaranteed to visit the locales in the order supplied by the GA-TSP but with flexibility in the time at which a locale is visited and with the same path smoothness properties. This is termed the GA-TSP PMHT algorithm.

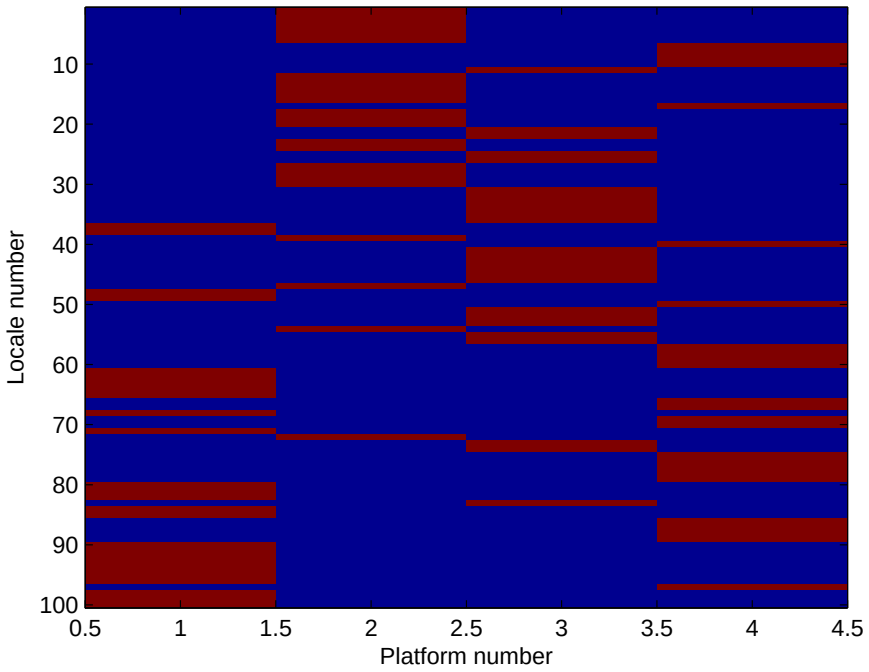
The GA-TSP provides a list of which locales are assigned to each platform which can be used to define the locale-to-platform assignment prior. The value of π^k_{nm} is set to unity for the platform assigned to locale n by the GA-TSP and zero for all others. Using the GA-TSP output example from Figure 4.9, the locale-to-platform assignment is shown in Figure 4.10(a).

The GA-TSP also provides an order in which to visit the locales. This order was used to define the locale-to-time assignments. The total batch length was divided into sub-batches and the order of locales was interpreted as an indication of which sub-batch the locale should be assigned to. The value of $\pi^\tau_{nt}(0)$ was uniform within the sub-batch identified by the GA-TSP and zero elsewhere. Using the GA-TSP output example once again, the locale-to-time assignment is illustrated in Figure 4.10(b).

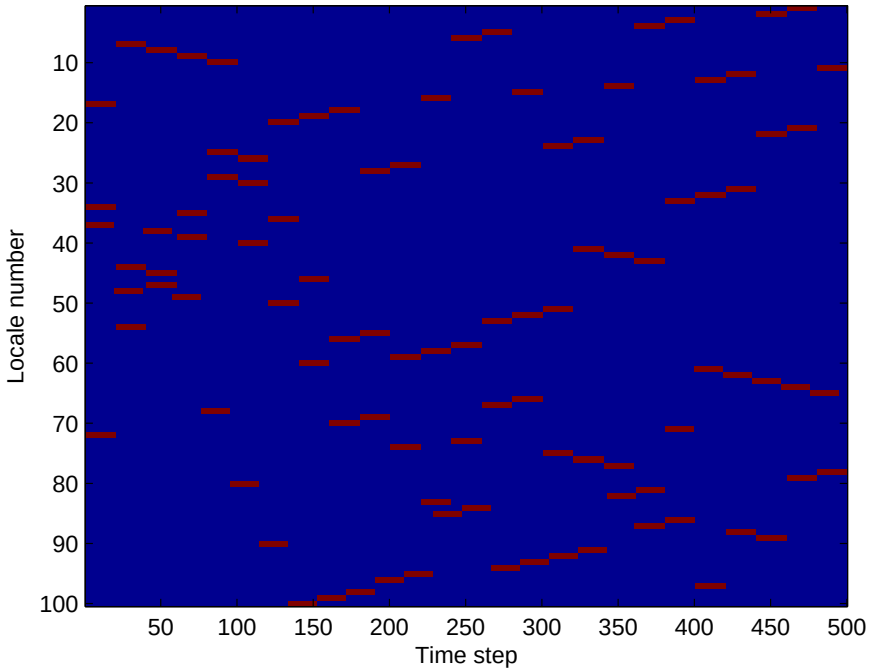
The resultant output from the GA-TSP after being PMHT-pp smoothed is shown in Figure 4.11. When compared with the GA-TSP output in Figure 4.9, there has been a tradeoff between smoother platform trajectories and the proximity to the locales. An interesting observation is that the PMHT-pp smoothing algorithm has maintained the crossover of the platforms. The performance of these two GA-TSP algorithms are compared with the PMHT-pp in the next section.

4.8.2 Path Planning Comparison

The use of the two algorithms for multiple platform trajectory planning is now illustrated through simulation. The area to be explored was a 10 unit square box. Two locale placement



(a) Initial value of π_{nm}^{k}



(b) Initial value of π_{nt}^{τ}

Figure 4.10: Initial priors for PMHT-pp smoothing

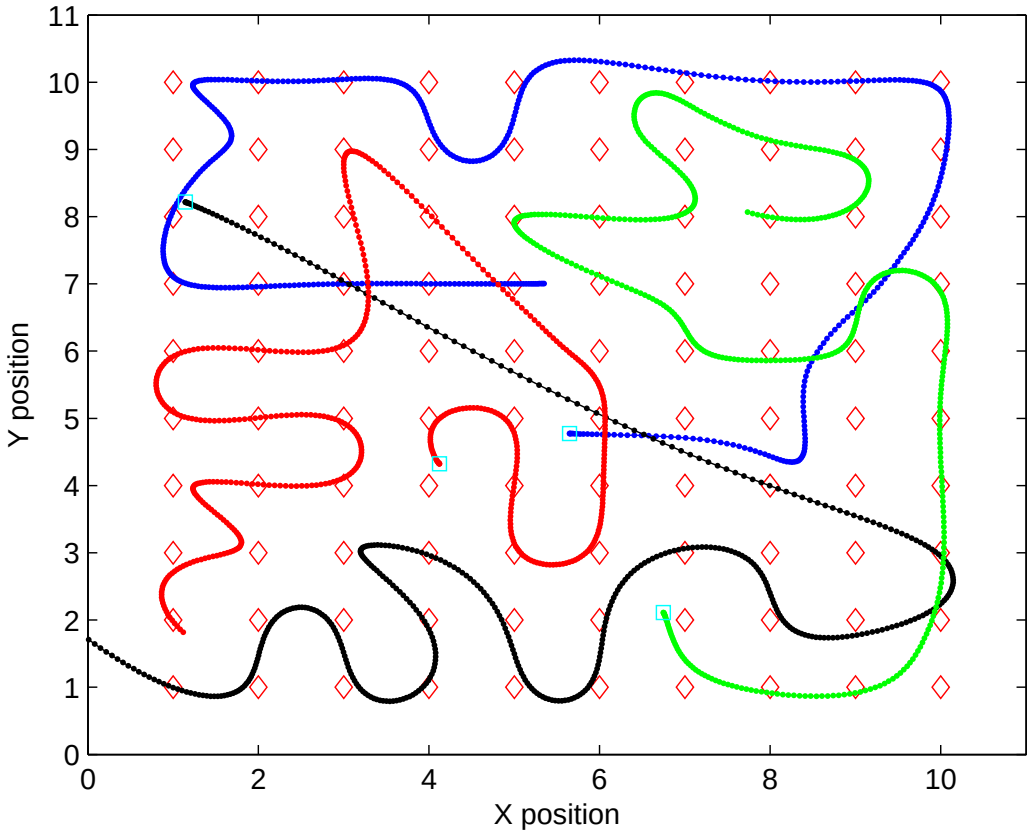


Figure 4.11: GA-TSP with PMHT-pp smoothed output

scenarios were considered: in the first, the locales were distributed over a uniform grid; in the second, the locales were distributed randomly with a uniform density. Fixed and randomised initial platform locations were also considered.

Three scenarios are presented here: a uniform grid of locales with randomised initial platform locations (A); randomised locale placement with fixed initial platform locations (B); and randomised locale placement and initial platform location (C).

The platform states were assumed to follow a constant acceleration model independently in X and Y as defined in Section 2.2.2. For the PMHT algorithm, the fitting penalty function was linear and Gaussian, with $Q = 10^{-8}I$ and $R = 10^{-2}I$.

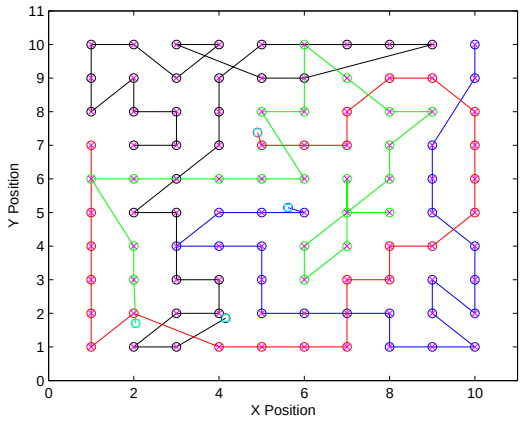
$T = 500$ time points were used. In each case, the initial state was assumed known, so the state estimates were initialised at the true position with zero speed and zero acceleration.

Figure 4.12 shows an example realisation of scenario A, which has a uniform grid of locales and random initial platform locations. The crosses mark the locations of the locales, the cyan boxes show the platform initial locations, and the coloured lines show the paths. With this scenario, the PMHT-pp divides the locales relatively evenly among the platforms with no crossovers whereas the GA-TSP has many platform crossovers, as does the GA-TSP PMHT smoothed algorithm.

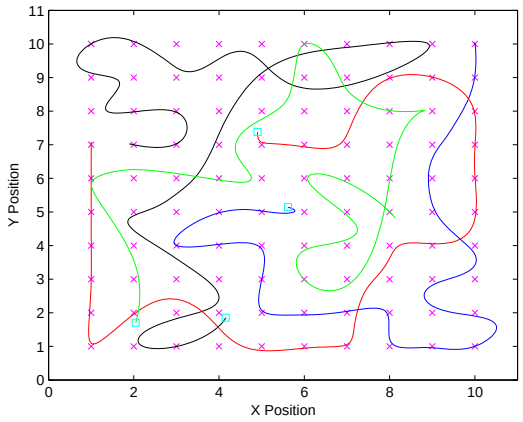
Figure 4.13 shows an example realisation of scenario B, with a three platform scenario where the platforms all began on the left side outside the map and the locales were randomly distributed. Again the PMHT-pp divides the space roughly evenly and the paths do not cross, whereas the TSP paths, for both versions, cross in several places. All algorithms tend to plan paths that pass through areas of high locale density and avoid areas where locales are sparse. Figure 4.14 shows an example of scenario C, where all four platforms had random initial positions in a field of randomly distributed locales.

Monte Carlo simulations of the above three scenarios were performed, and the total distance travelled by the platforms averaged over 100 trials. These average total distances and the CPU time incurred for each algorithm are given in Table 4.3. Clearly the PMHT-pp paths are significantly shorter. The relationship between the GA-TSP paths and the GA-TSP PMHT smoothed paths is less direct. Two factors influence the path length in the smoothed case: the path may need to be extended in order to turn a sharp corner, but the platform is no longer required to pass exactly over the locale. These two factors have opposing effect and the net result depends on the scenario.

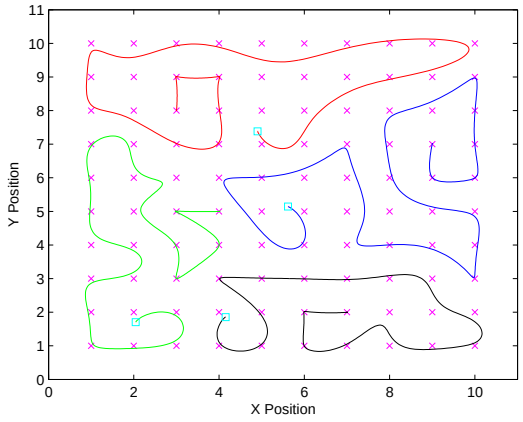
There is a clear advantage of the PMHT-pp algorithm compared with the GA-TSP. A possi-



(a) GA-TSP

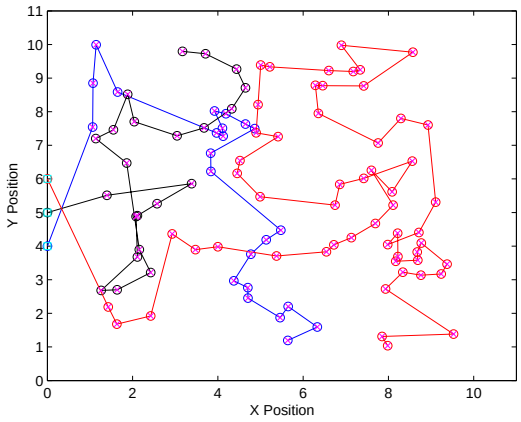


(b) GA-TSP PMHT smoothed

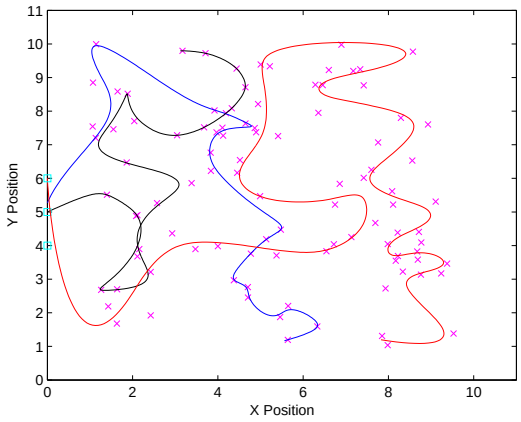


(c) PMHT-pp

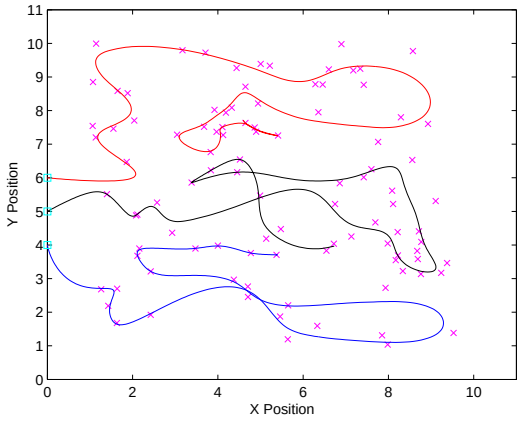
Figure 4.12: Results for 4 platforms and grid of locales



(a) GA-TSP

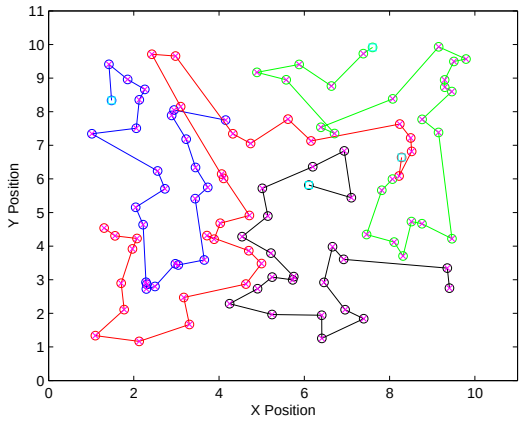


(b) GA-TSP PMHT smoothed

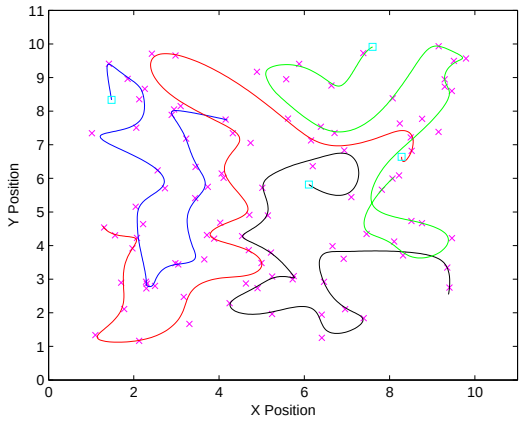


(c) PMHT-pp

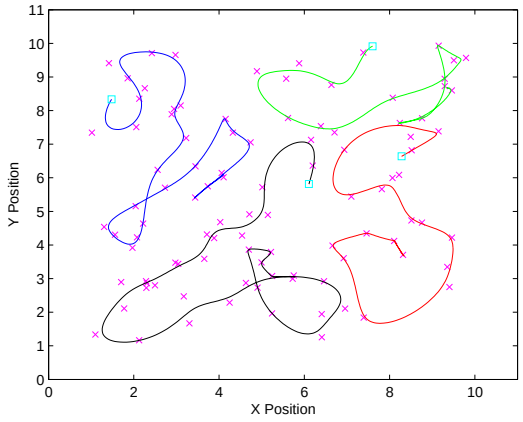
Figure 4.13: Results for 3 platforms and random locales



(a) GA-TSP



(b) GA-TSP PMHT smoothed



(c) PMHT-pp

Figure 4.14: Results for 4 platforms and random locales

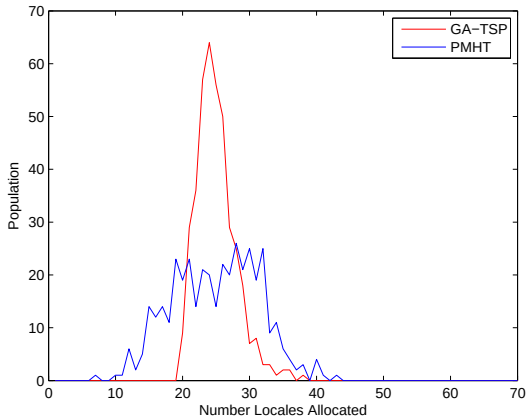
Table 4.3: Monte Carlo Comparison results

	A	B	C
Total Distance			
GA-TSP	117.8	118.8	87.8
GA-TSP PMHT smoothed	124.6	93.8	92.8
PMHT-pp	94.6	77.8	78.8
CPU time (s)			
GA-TSP	18.3	17.4	18.0
GA-TSP PMHT smoothed	39.4	33.3	38.9
PMHT-pp	23.1	16.7	22.8

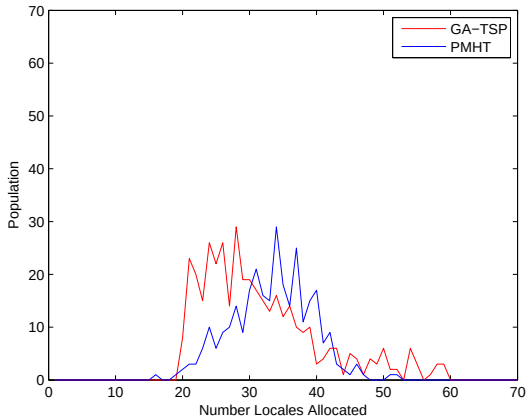
ble reason why the distance travelled is lower is that the PMHT-pp path is allowed to kill two birds with one stone: under the PMHT-pp model, it is permissable to assign more than one locale to a platform at a particular instant. This allows the PMHT-pp path to pass between two locales and cover them both at the same time, provided that they are close enough together. This is not permitted by the TSP approach. In the context of surveillance, it is quite feasible that the sensor may be able to observe the area designated by more than one locale in the same frame, so the PMHT-pp behaviour is appropriate provided that the locale fitting function adequately reflects sensor coverage.

Table 4.3 also lists the CPU time incurred by each of the algorithms and it shows that the PMHT takes a similar amount of time as the GA-TSP to process. Various parameters can be tuned in either algorithm to decrease CPU time, such as the number of iterations allowed for convergence. As expected, the average smoothed TSP CPU time is approximately the sum of the TSP and the PMHT CPU times.

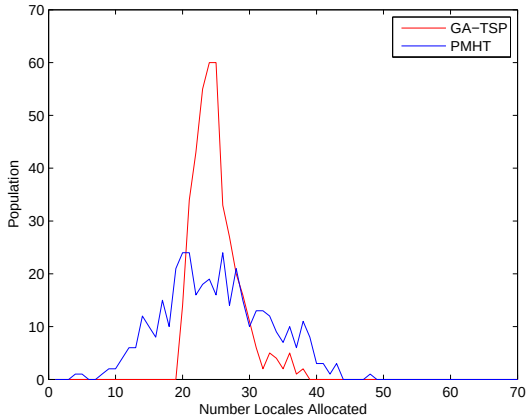
Figure 4.15 shows a histogram of the number of locales allocated to each platform, averaged over the Monte Carlo trials. For the fixed initial platform scenario (scenario B), this provides little intuition, but for the other two scenarios, the GA-TSP variance is much smaller than the PMHT variance: it would appear that the GA prefers to find solutions that allocate a similar number of locales to each platform, whereas the PMHT tends to be less restrictive.



(a) Scenario A



(b) Scenario B



(c) Scenario C

Figure 4.15: Locale distribution between platforms

4.9 Sliding batch PMHT Path Planning

The PMHT examples shown so far have computed the paths for the multiple platforms for all time points in one go, essentially calculating all the associations in one large batch. This section will briefly explore the advantages of using a smaller batch in a sliding window format to perform path planning.

An immediate benefit of a sliding window batch is that the list of locales of interest may change as the platforms follow their assigned routes. The sliding window allows for updates to the paths, so at a set time, the platforms may replan the rest of their routes. Depending on where the platforms may have already visited, the new plan may include reassigning locales to a different platform or altering the visit order.

The ability to revisit an area already visited by another platform is a key objective in SLAM. As SLAM is essentially a method to incrementally self localise, the platform position estimation errors increase with the exploration of an unknown area. By having the platforms revisit previously mapped landmarks, it is possible to reduce the uncertainty in the built map. This concept is called “closing the loop” and is the focus of much research [RNDW07][NCH06][WCN+09].

The strategy of the sliding window batch version depends on the overall goal. The general strategy is that initially, all the locales of interest are assigned to the platforms so that the platforms have a full trajectory to cooperatively traverse all the locales.

At certain time steps, the paths are replanned for each of the platforms. The locales used in this replanning step can be either the locales that have not been visited yet, or a weighted list of the locales where unvisited locales have a higher priority to be visited over the already visited locales. This is then repeated at future time steps.

If the locales have all been visited, it is possible that there will be no more locales for the path planner. In this case, the process is either considered complete or locales can be added again for a revisit. In consideration of SLAM, the landmarks that were detected and tracked for the map could be used as new locales to revisit.

Figure 4.17 gives an example of the sliding window batch. In this example, there are four platforms initialised at random positions in a grid of size 10 by 10. The size of the batch is 500 time steps and the size of the overlap is 250 time steps. At iteration 1, the full trajectory is planned for the four platforms. At iteration 2, the platforms have moved over the first 250 time steps of their trajectories. The next 500 time steps of their trajectories are replanned. This process is repeated for 8 iterations in total. The red squares mark the position of the platforms

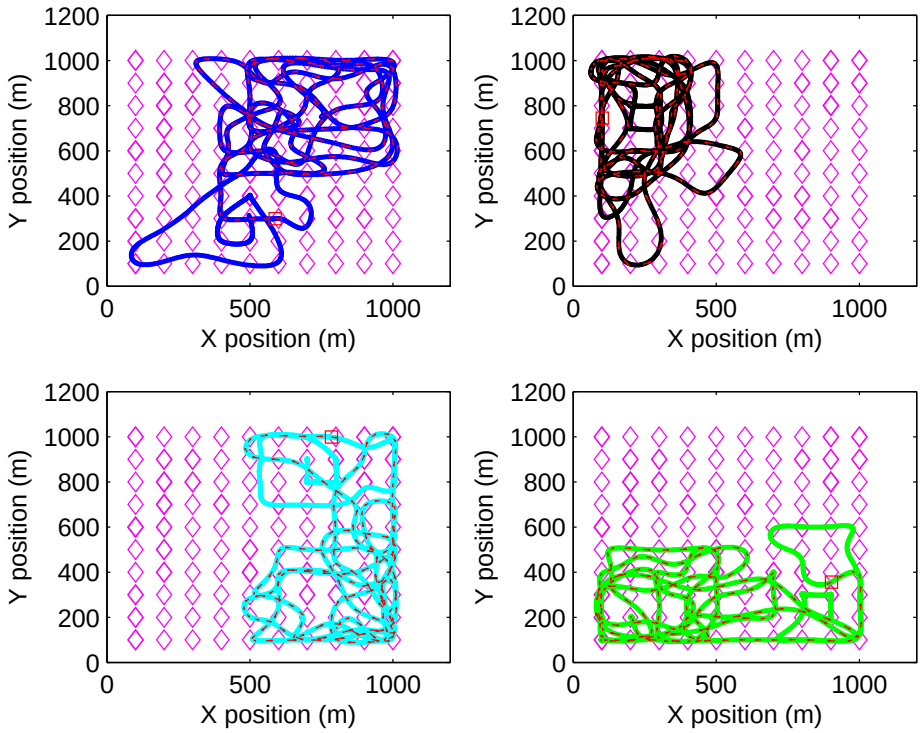


Figure 4.16: Trajectories separated for the four platforms after 8 iterations

at the start of each new batch.

At the end of each iteration, it is determined which locales have been visited by the platforms. No locales are removed from the set, but locales that have not been visited are given twice the weight compared to locales that have been. If a locale has not been visited after another duration, the weight is doubled again.

Figure 4.16 shows the full trajectories planned for each of the platforms over all 8 iterations. It is observed that the platforms have moved and covered locales in other regions and that the tracks overlap. Because locales that had been visited were still in the list of locales, there was revisiting of locales.

The sliding batch PMHT-pp has demonstrated a method to replan the platform paths when necessary. This technique has many benefits such as revisiting an area, encouraging crossing paths, decreasing localisation errors and linking submaps together in the case of a decentralised system.

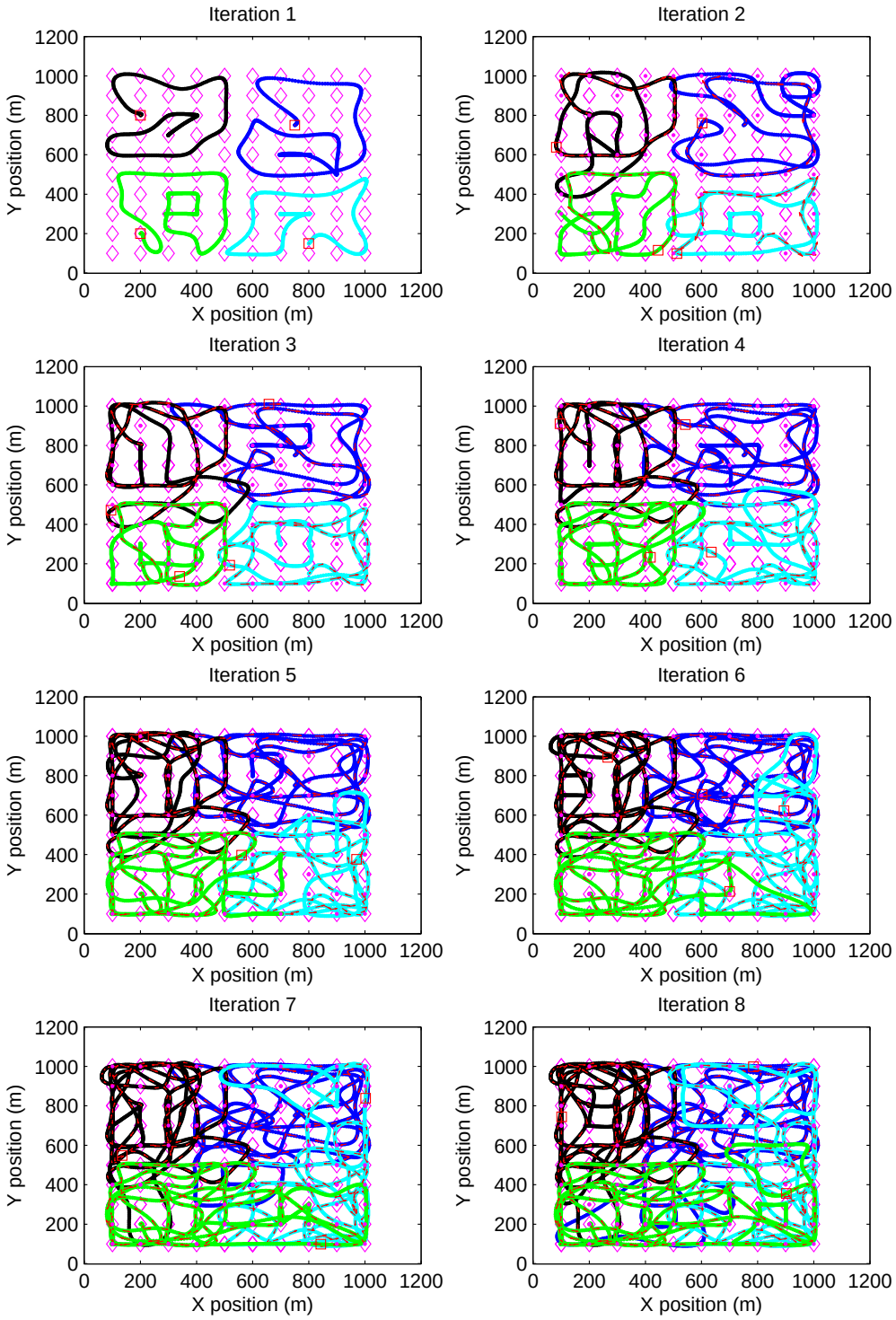


Figure 4.17: Trajectories at each iteration

4.10 PMHT-pp with a Particle Filter Estimator for Indoor Environments

The applications of PMHT-pp have so far assumed linear Gaussian models for the target dynamics. For many applications, the assumption of linearity is not valid, but the non-linearity can be mitigated through use the of an Extended Kalman Filter, or an Unscented Kalman Filter. Examples of this are given in the next chapter. However, these methods still assume that the target posterior pdf can be approximated as a Gaussian. One example where this is not an appropriate model is when there are hard constraints on the target motion. For example, in an indoor environment, walls present physical barriers through which motion is impossible. An appropriate way to model this is to truncate the pdf, however, this truncation is not compatible with Kalman based filters. Instead, a particle filter should be used. An example of PMHT using particle filtering can be found in [Dav11]. This section demonstrates how a particle filter can be integrated with PMHT-pp for constrained targets. Recall in Section 2.2.7 from Chapter 2, the PF allows non-linear estimation of the platform states and by estimating a non-Gaussian distribution, the platform trajectories can be created around walls and obstacles. Section 4.10.1 describes the implementation of the PMHT-pp-pf algorithm and the performance is illustrated with some simple examples in Section 4.10.2.

4.10.1 PMHT-pp-pf Indoor Implementation Strategies

In the implementation of the PMHT-pp with a particle filter estimator (denoted the PMHT-pf-pp algorithm), a set of particles is generated for each of the initial platform states. The sets of particles are propagated forward using the platform motion models with additive random Gaussian process noise.

The associated particle weights are determined based on the distance between the particle position and the synthetic measurements produced by the PMHT-pp, given in Section 4.3.

The advantage of using particle filtering is the set of particles can be weighted to control the motion of the platforms. An example is in an indoor environment with walls that the platforms must traverse around. Given a set of locales of interest to visit and a map of walls the platforms cannot pass through, the PMHT-pf-pp can use the particle filter to estimate trajectories that the standard linear PMHT-pp cannot produce.

In the case where the proposed trajectory would have the particle propagating through a

wall, the associated weight of the particle, w_t^i is set to 0 and is generally removed during the resampling step. With this strategy, the particle filter will iterate towards a final trajectory that can work around walls.

The walls not only obstruct the movement of the platforms, but they may also obstruct the platforms from observing the landmarks on the opposite side of the wall. This is modelled by setting the PMHT weight associations to 0 if a particular locale is on the other side of a wall.

Another advantage of the particle filter is that it can be used where the target model or penalty function are non-linear and non-Gaussian. In this problem, the particle filter provides a great advantage for the estimation of the platform's non-linear motion.

The target trajectories are estimated using the weighted mean of the particles using equation (2.45). This does not follow the Maximisation step of the EM algorithm strictly, but provides a sufficient estimate of the target state approximated to the accuracy permitted by the number of particles used.

4.10.2 PMHT-pp-pf Indoor Results

The use of the particle filter with PMHT for multiple platform trajectory planning is now illustrated through simulation. The area to be explored is a 10 unit square box. A 10×10 uniform grid of locales was defined and the PMHT-pp-pf algorithm described in the previous section was used to plan paths. Various combinations of initial platform position, number of platforms and wall mappings were considered and a selection are presented here.

The platform states were modelled with a state vector consisting of its 2-D position and its motion, which was approximated using a constant speed and constant turn rate model as in Section 2.2.2,

$$x_t^m \equiv \left\{ \begin{array}{c} \text{X position} \\ \text{Y position} \\ \text{heading} \\ \text{speed} \\ \text{steering angle} \end{array} \right\}_t \tag{4.25}$$

with white Gaussian process noise on the speed and steering angle to account for manoeuvres.

The state evolution process was defined by

$$p(x_t^m | x_{t-1}^m) = \mathcal{N}\left(x_t^m; F(x_{t-1}^m), Q\right), \tag{4.26}$$

The process noise was estimated by the set of particles with Q having mean values

$$Q = \begin{bmatrix} 0.05 \\ 0.05 \\ 0.001 \\ 0.00001 \\ 0.000001 \end{bmatrix} \tag{4.27}$$

The fitting penalty function was linear and Gaussian,

$$\zeta(z_n|x_t^m) = \mathcal{N}\left(z_n; Hx_t^m, R\right), \tag{4.28}$$

with $H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$ and $R = 10^{-2}I$.

$T = 500$ time points were used. In each case, the platforms were initialised at a fixed position with zero speed and turning rate. 500 particles were used in the particle filter. This number was a tradeoff between the speed and the accuracy of the results of the algorithm.

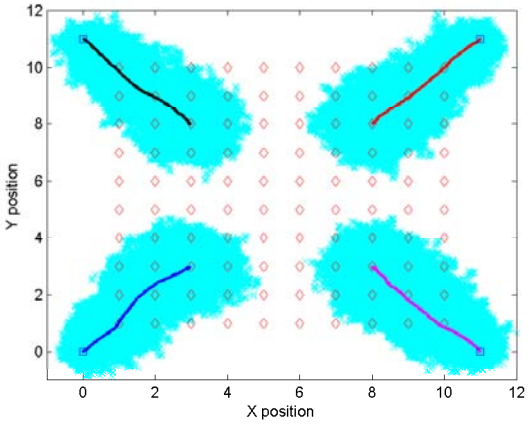
For the following results, the trajectory of each platform is represented by a coloured line, the initial platform location with a blue square, locales with a red diamond and the particles with a cyan cloud.

Figure 4.18 shows the evolution of the trajectories for four platforms after a number of EM iterations for a grid of locales. Initially, the assignment implicitly divides the area of landmarks into quadrants and each platform moves in a straight line towards the center due to the symmetry. However, after a period, the paths diverge and eventually span each quadrant. The constraints on the platform dynamics inherent in the process model result in smoothly curving paths.

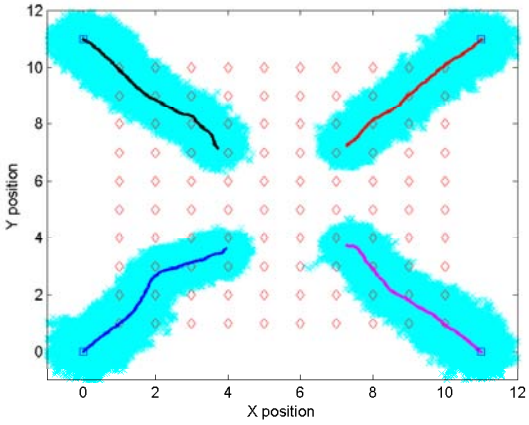
It is observed that the variance spread of the particles decreased over the duration of the trajectory. This is caused by the resampling of the particles at each time with the highest weighted particles dominating.

Figure 4.19 directly compares the converged output from the previous PMHT-pp simulation result with that of the PMHT-pp-pf. Although the area has been divided evenly into quadrants, the platform paths are very different between the two algorithms. While the PMHT-pp algorithm in Figure 4.19(a) has the trajectories forming loops to cover each quadrant, the PMHT-pp-pf in Figure 4.19(b) has planned trajectories that “zigzag” along the diagonal to reach the locales.

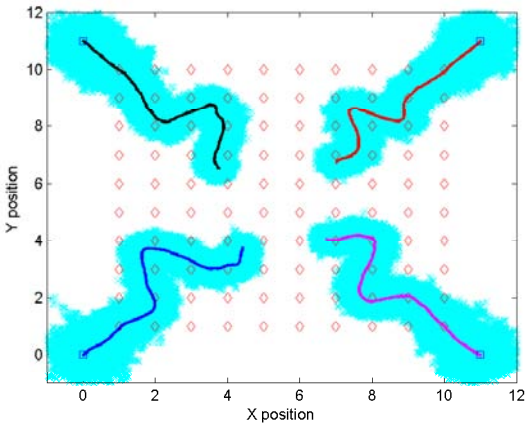
The remaining scenarios model an indoor environment with walls which are represented



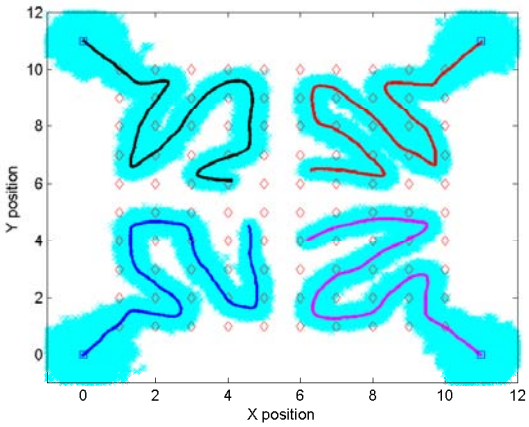
(a) Iteration=1



(b) Iteration=3

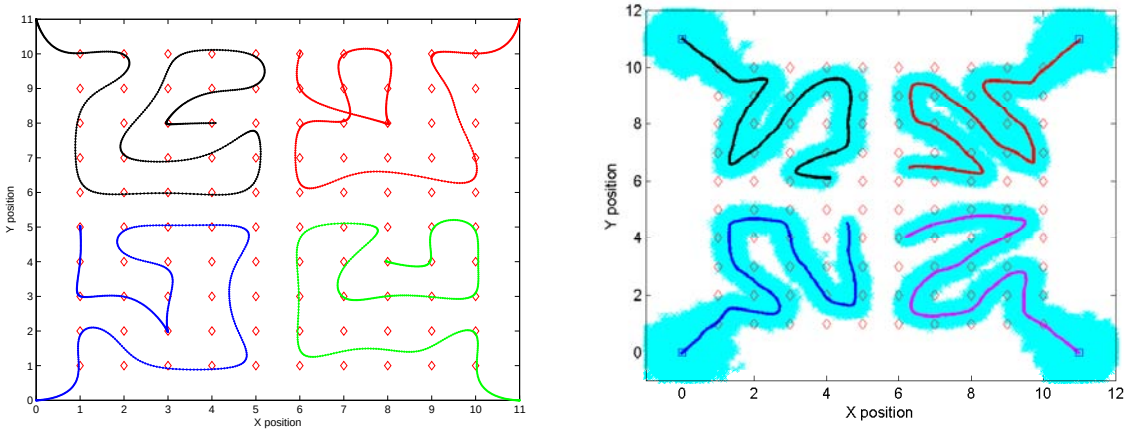


(c) Iteration=5



(d) Converged at Iteration=10

Figure 4.18: Assigned trajectories evolution for 4 platforms with grid of locales



(a) Converged PMHT-pp (b) Converged PMHT-pp-pf

Figure 4.19: Comparison between PMHT-pp and PMHT-pp-pf with a grid of locales

with black lines in the following figures.

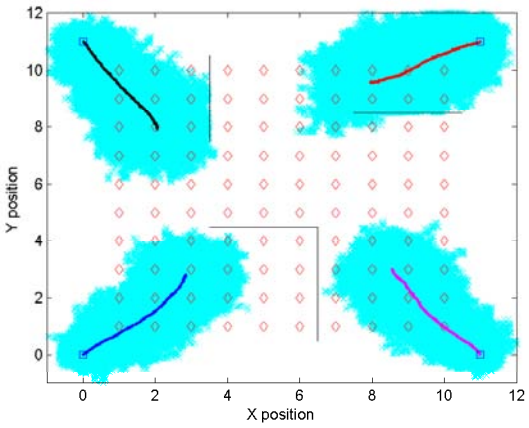
Figure 4.20 show a scenario where there are four walls placed in the area that obstruct the trajectories of the platforms. Figure 4.20 shows the results for a uniform grid of locales with four platforms once again initalised in the outer four corners of the area.

Similar to the previous scenario, the four platforms attempt to carve the area evenly and move in a straight line toward the middle. It is observed that due to the particles that transition through a wall are given a weight of 0, the cloud of particles do not pass through walls and tend to wrap around the wall. This formation of particles produces a platform trajectory that can manoeuvre around the walls.

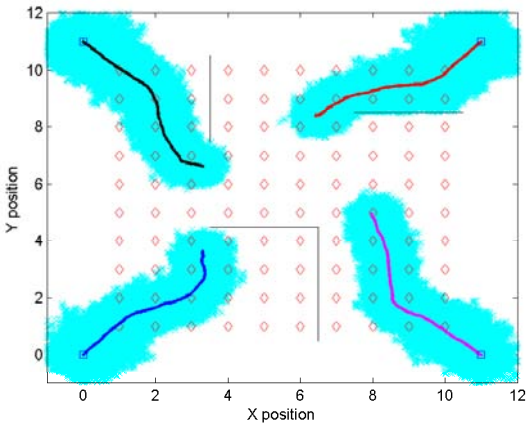
The final scenario takes the form of a building with two rooms on each side and a larger room at the back. Figure 4.21 shows the evolution of the platform trajectories through the indoor structure. Two platforms initialised at the bottom of the building are tasked with exploring the area with a grid of locales.

As observed in Figure 4.21(b), the platform trajectories may inadvertently traverse through a wall. This is caused by the spread of the particle cloud covering both sides of the wall. Even though the individual particles have been constrained to not transition through a wall, the weighted mean of the particles may still produce such a trajectory. This problem is generally corrected with further iterations as the particles converge.

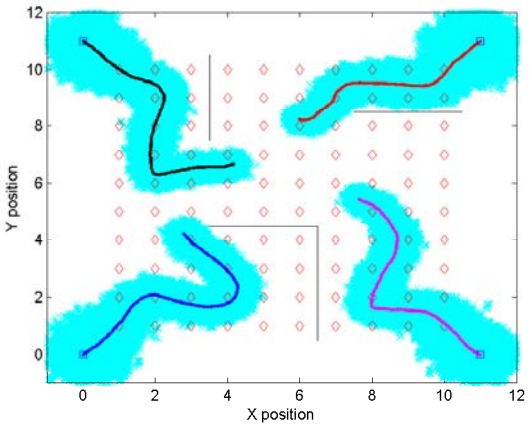
Another interesting observation is that due to the doorway structure, the particle cloud may



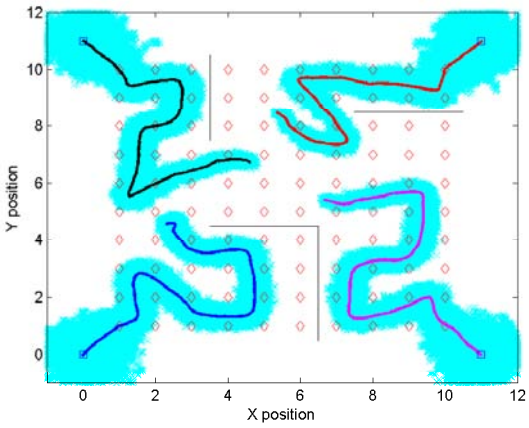
(a) Iteration=1



(b) Iteration=3

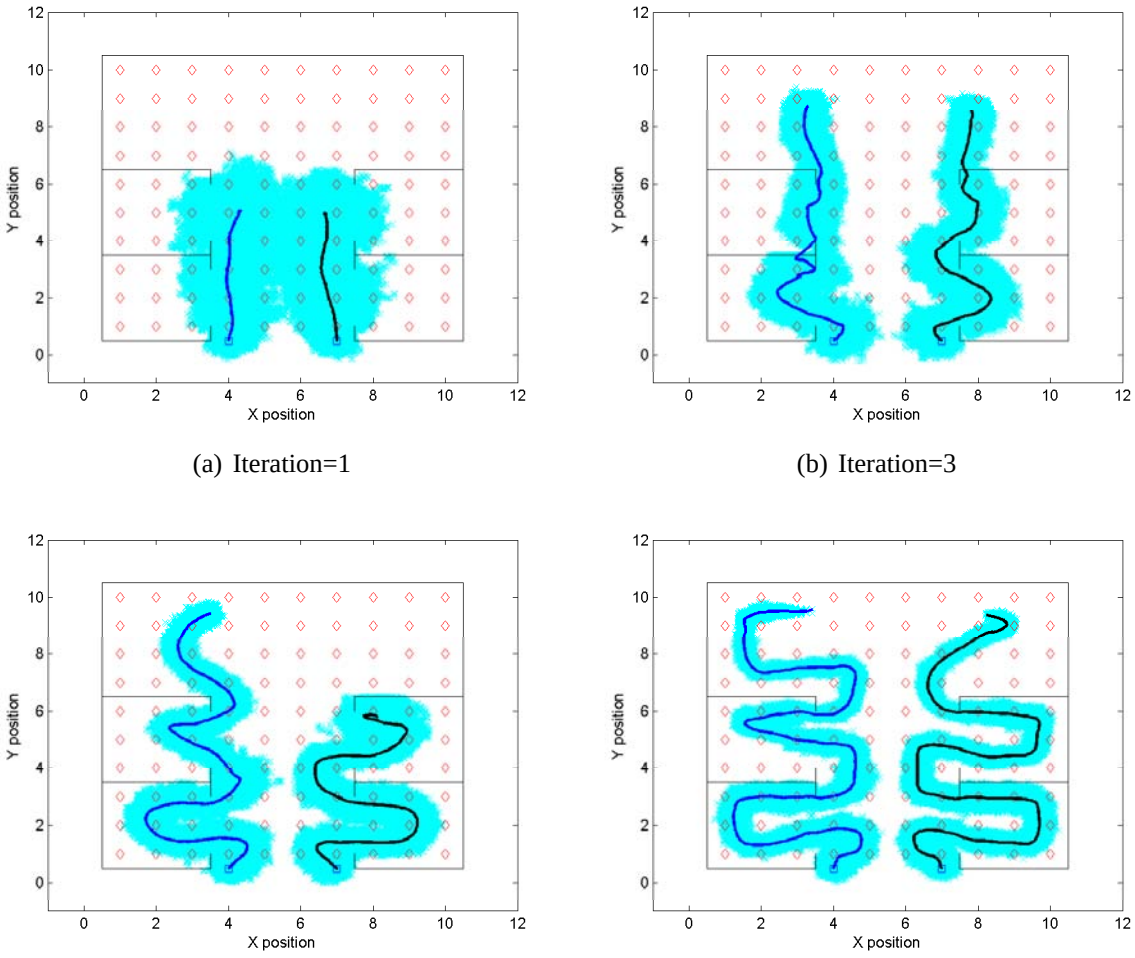


(c) Iteration=5



(d) Converged at Iteration=10

Figure 4.20: Example assigned trajectories evolution for 4 platforms with four walls within a grid of locales



(a) Iteration=1

(b) Iteration=3

(c) Iteration=5

(d) Converged at Iteration=10

Figure 4.21: Assigned trajectories evolution for 4 platforms within an indoor environment with a grid of locales

get stuck in a corner as shown in Figure 4.21(c). Once again, with further iterations, the trajectory was corrected.

4.11 PMHT-pp with Non-Homogeneous Locales

The previous sections described the path planning algorithm as being guided by a set of discrete locales, either deterministically placed in a grid or randomly placed by sampling a uniform spatial distribution. This section generalises the path planning in two ways. The first is in Section 4.11.1, which demonstrates how to incorporate a non-homogeneous (spatially non-uniform) set of locales based on a model of priority. Secondly, Section 4.11.2 generalises the non-homogeneous discrete set of points to an arbitrary intensity function over a continuous valued spatial area. This is reminiscent of the problem in Section 4.7, but with a non-homogeneous intensity. This resulting improvement to the PMHT path planner no longer uses the notion of artificially generated locales to guide the platform trajectories, but instead it is driven by an intensity function. Finally, a simulation example of this extension is shown in Section 4.11.3.

4.11.1 Non-Homogeneous Locales

Suppose a particular locale, s was more important than the others. This means that it would be preferable if the output of the path planning algorithm placed higher emphasis on passing close to the location s than other locations in the search space. There are two obvious ways in which the path planner could be influenced to achieve this. The first is to place duplicate locales at s. If there is more than one locale then this will have the same effect as collecting two sensor observations at a certain location: the resulting state estimate will be closer to s than if there was only one. Alternatively, the fitting covariance R could be scaled for locale s, analogous to the assertion that this locale is more accurate than the others. The approach detailed here will follow the first method.

Suppose that a certain locale s is duplicated, so that there are now $N+1$ locales, $[1 \dots N] \cup s$. This does not change the association weights, since they are independent for each locale. What changes is the range of the summand in the locale contribution to Q_X in (4.18). The effect of this is to duplicate the term due to locale s in the synthetic measurement and covariance, namely

(4.20) becomes

$$\begin{aligned} z_t^m &= \frac{\sum_{n=1}^{N+1} w_{ntm} z_n}{\sum_{n=1}^{N+1} w_{ntm}} \\ &= \frac{\left(\sum_{n=1}^{N} w_{ntm} z_n\right) + w_{stm} z_s}{\left(\sum_{n=1}^{N} w_{ntm}\right) + w_{stm}}, \end{aligned}$$

Similarly for the synthetic covariance in (4.21) becomes

$$\tilde{R}_t^m = \frac{1}{\left(\sum_{n=1}^{N} w_{ntm}\right) + w_{stm}} R.$$

The state estimation proceeds as previously and the resulting path will be influenced to pass closer to z_s.

The intuitive extension of this is now to place an integer number of copies of each locale, η_n, with a higher value of η_n where the locales have higher priority. As before, each of these duplicates has the same weight as z_n, so the synthetic measurement (4.20) and covariance (4.21) become

$$\tilde{z}_t^m = \frac{\sum_{n=1}^{N} w_{ntm} \eta_n z_n}{\sum_{n=1}^{N} w_{ntm} \eta_n}, \tag{4.29}$$

$$\tilde{R}_t^m = \frac{1}{\sum_{n=1}^{N} w_{ntm} \eta_n} R. \tag{4.30}$$

Note that by introducing the priority η_n the effect is to reduce the synthetic covariance $\tilde{R}_t^m$ since $\sum_{n=1}^{N} w_{ntm}\eta_n > \sum_{n=1}^{N} w_{ntm}$ this can be compensated for by using a larger initial value of R, although this will influence the weights.

4.11.2 PMHT-pp with Priority as a Continuous Density

Although the variable η_n represents a count of the number of duplicates of z_n the resulting synthetic locale and covariance in (4.29) and (4.30) could be evaluated for non-integer values of η_n. Accordingly, η_n is now referred to as the *priority* of locale n and allowed to take on a positive semi-definite continuous value. This could be achieved rigorously by defining $\bar{\eta}_n = \lfloor \eta_n / \hbar^2 \rfloor$ and taking the limit as $\hbar^2 \to 0$ in the same manner as the derivation of Histogram-PMHT

[Str00].

Suppose now that the exploration area, $\mathcal{A}$, is a region in the plane and there is a known continuous priority intensity, $\rho(z) \geq 0 \forall z \in \mathcal{A} \subset \Re^2$.

Define a uniform grid of N pixels in the exploration area with pixel size Δ^2. Let the center of pixel n be z_n and the priority of pixel n be $\eta_n = \Delta^2 \rho(z_n)$. Clearly η_n is a piecewise constant approximation to $\rho(z)$. If $\rho(z_n)$ is the intensity of a Poisson point process, then η_n is the expected number of locales in the area of pixel z_n. Further, $\{\eta_n, z_n\}$ could be interpreted as a sample approximation of the intensity

$$\rho(z) \approx \sum_{n=1}^{N} \delta(z - z_n)\eta_n, \tag{4.31}$$

where $\delta(z)$ is the Dirac delta function.

The weight notation is redefined from w_{ntm} to $w_{tm}(z_n)$, which gives the ability to represent the weights for any $z \in \mathcal{A}$, i.e.

$$w_{tm}(z) = \frac{\hat{\pi}^k_{nm}\hat{\pi}^\tau_{nt}\zeta(z|\hat{x}^m_t)}{\sum_{r=1}^{T}\sum_{s=1}^{M}\hat{\pi}^\tau_{nr}\hat{\pi}^k_{ns}\zeta(z|x^s_r)}. \tag{4.32}$$

The advantage in using this continuous valued η_n is that priorities can be balanced between locales without leading to a deflation of $\tilde{R}^m_t$.

Consider now the state dependent term from the auxiliary function, i.e. Q_X given by the modification of (4.18) to include the priority

$$\begin{aligned} Q_X &= \log P(X) + \sum_{n=1}^{N}\sum_{t=1}^{T}\sum_{m=1}^{M}\eta_n w_{tm}(z_n)\log\zeta(z_n|x^m_t), \\ &= \log P(X) + \sum_{t=1}^{T}\sum_{m=1}^{M} q_{tm}. \end{aligned} \tag{4.33}$$

where q_{tm} represents the locale contribution to the auxiliary function for platform m at time t. Recalling that ζ is a multi-variate Gaussian with mean Hx^m_t and covariance R, so

$$\log\zeta(z_n|x^m_t) = \mathsf{const} - \frac{1}{2}(z_n - Hx^m_t)^\mathsf{T} R^{-1}(z_n - Hx^m_t), \tag{4.34}$$

where the constant term is due to the normalising term in the Gaussian pdf and is common for all locales and platforms.

Equation (4.34) is substituted into (4.33) and the limit of the auxiliary function is taken as $\Delta^2 \to 0$. It suffices to consider each q_{tm} individually as shown in equations (4.35) through (4.38) below. The constant term in these equations changes from one step to the next but contains only irrelevant terms that are independent from the target state.

$$\lim_{\Delta^2 \to 0} q_{tm} = \lim_{\Delta^2 \to 0} \sum_{n=1}^{N} \eta_n w_{tm}(z_n) \Big\{ \mathsf{const} - \frac{1}{2}(z_n - Hx_t^m)^{\mathsf{T}} R^{-1} (z_n - Hx_t^m) \Big\} \tag{4.35}$$

$$\begin{aligned}
&= \mathsf{const} - \frac{1}{2} \lim_{\Delta^2 \to 0} \sum_{n=1}^{N} \Delta^2 \rho(z_n) w_{tm}(z_n) \\
&\qquad \Big\{ (z_n)^{\mathsf{T}} R^{-1} z_n - 2(Hx_t^m)^{\mathsf{T}} R^{-1} z_n + (Hx_t^m)^{\mathsf{T}} R^{-1} (Hx_t^m) \Big\} \\
&= \mathsf{const} + (Hx_t^m)^{\mathsf{T}} R^{-1} \Big\{ \lim_{\Delta^2 \to 0} \sum_{n=1}^{N} \Delta^2 \rho(z_n) w_{tm}(z_n) z_n \Big\} \\
&\qquad - \frac{1}{2} \Big\{ \lim_{\Delta^2 \to 0} \sum_{n=1}^{N} \Delta^2 \rho(z_n) w_{tm}(z_n) \Big\} (Hx_t^m)^{\mathsf{T}} R^{-1} (Hx_t^m) \\
&= \mathsf{const} + (Hx_t^m)^{\mathsf{T}} R^{-1} \Big\{ \int_{\mathcal{A}} \rho(z)\ w_{tm}(z)\ z\ dz \Big\} \\
&\qquad - \frac{1}{2} \Big\{ \int_{\mathcal{A}} \rho(z)\ w_{tm}(z)\ dz \Big\} (Hx_t^m)^{\mathsf{T}} R^{-1} (Hx_t^m) \qquad (4.36) \\
&= \mathsf{const} + (Hx_t^m)^{\mathsf{T}} \left(\tilde{R}_t^m \right)^{-1} \tilde{z}_t^m - \frac{1}{2} (Hx_t^m)^{\mathsf{T}} \left(\tilde{R}_t^m \right)^{-1} (Hx_t^m) \qquad (4.37) \\
&= \mathsf{const} - \frac{1}{2} (\tilde{z}_t^m - Hx_t^m)^{\mathsf{T}} \left(\tilde{R}_t^m \right)^{-1} (\tilde{z}_t^m - Hx_t^m) \qquad (4.38)
\end{aligned}$$

The progression from summation to integral in (4.36) follows the Riemann notion of integration, which is sufficient for the types of functions expected for $\rho(z)$ and $w_{tm}(z)$. The new synthetic locale and covariance implicitly defined by (4.37) and (4.38) are given by

$$\tilde{z}_t^m = \frac{\int w_{tm}(z)\ \rho(z)\ z\ dz}{\int w_{tm}(z)\ \rho(z)\ dz} \tag{4.39}$$

$$\tilde{R}_t^m = \frac{1}{\int w_{tm}(z)\ \rho(z)\ dz} R. \tag{4.40}$$

Although this formulation was reached by taking the limit of a uniform grid of locales, the same result could be obtained for a less regular configuration in the same way that Riemann integration need not be derived from a set of intervals with equal support. Using something other than the uniform grid would result in a locale-dependent Δ_n^2, but the limit as all of these approach zero would be the same.

The resulting algorithm no longer relies on a set of artificial points in the exploration region to guide the trajectories. Moreover it even provides a satisfying interpretation of the original method: the discrete grid of locales used to define the exploration region in the previous sections can be seen to be a numerical approximation to the path planning solution for a uniform priority intensity map. It is noted that the integrals in (4.39) and (4.40) do not appear to be tractable since $w_{tm}(z)$ is the ratio of a Gaussian to a sum of Gaussians. So it is likely that a numerical integral approximation would need to be used resulting in essentially the algorithm in Section 4.11.1.

The result described above assumes a constant priority intensity function, $\rho(z)$. An important extension would be to make $\rho(z)$ time-varying and dependent on the target state. This then would allow the intensity function to actually represent the expected number of targets in a region, conditioned on the sensor data, i.e. the Probability Hypothesis Density [Mah03]. In this case, the path planning solution could solve a much more general sensor resource scheduling problem.

4.11.3 Simulation Example

The difference between the existing PMHT-pp method presented in Section 4.4 and the new method derived in this section is now illustrated through a simple simulation. The new method is denoted as the non-homogeneous PMHT-pp.

Assume that the exploration area $\mathcal{A}$ is a 10 unit square box. The PMHT-pp method guides a single platform through the area by constructing a 10×10 uniform grid of discrete locales. Whereas the non-homogeneous PMHT-pp is guided by an intensity function that is a sum of

two Gaussian components on a uniform pedestal,

$$\rho(z) = 0.01 + \sum_{j=1}^{2} \exp\Big\{ -\frac{1}{2}(z-\mu_j)^{\mathsf{T}}\Sigma_j^{-1}(z-\mu_j)\Big\}, \tag{4.41}$$

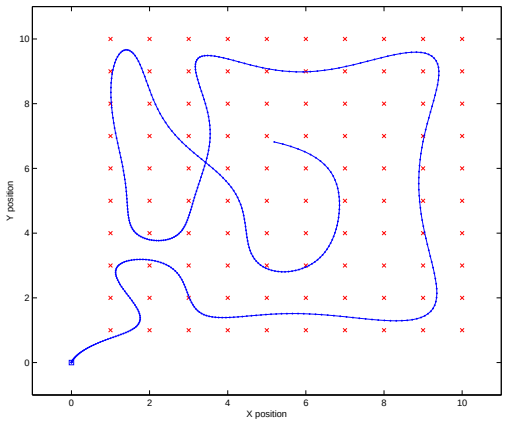
with $\mu_1 = [8,3]^{\mathsf{T}}$, $\Sigma_1 = 0.25I$, $\mu_2 = [4,8]^{\mathsf{T}}$ and $\Sigma_2 = \mathsf{diag}(16, 0.8)$. The priority intensity contains an isolated point of high priority represented by the first component and a narrow ridge of high priority represented by the second.

A third method using a set of non-uniformly spatially distributed locales of equal weight to represent the continuous priority intensity from (4.41) is also compared. The PMHT-pp method was used to guide a single platform through a hundred locales which were randomly placed by sampling the priority intensity function.

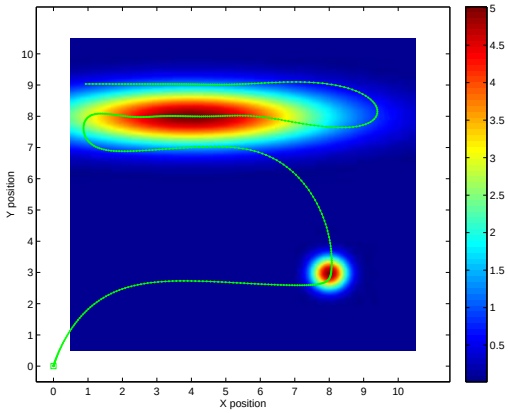
The platform state was assumed to follow a constant acceleration model independently in X and Y as defined in Section 2.2.2, with $Q = 10^{-8}I$. The fitting penalty (analogous measurement probability density) function was linear and Gaussian, with $H = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$ and $R = 10^{-2}I$.

$T = 500$ time points were used. The platform was initialised at position $[0,0]$ with zero speed and zero acceleration.

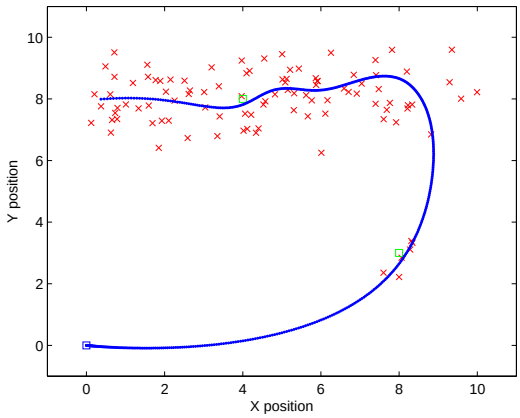
Figure 4.22(a) shows the output of the discrete-locale PMHT-pp over the uniform grid. A trajectory is created that spans the grid of locales. Since the algorithm is not aware of the higher priority regions, the path pays no particular attention to them. In particular, the platform does not closely visit the area around $[8,3]$. Figure 4.22(b) shows the output of the non-homogeneous PMHT-pp overlaid on the priority intensity map, as represented by (4.41). Clearly the paths in Figures 4.22(a) and 4.22(b) are very different. The non-homogeneous PMHT-pp focuses on the locations of high priority and quickly skirts over the remaining area. Because the variance of the first intensity component is so small, the platform trajectory passes directly through its centre. The path then makes several passes over the extended component. Figure 4.22(c) shows the output of the PMHT-pp using a set of non-uniformly spatially distributed locales of equal weight to represent the priority intensity map. The platform visits both areas of high priority, travelling through the centre of each region of locales.



(a) Discrete-locale PMHT-pp trajectory



(b) Non-Homogeneous PMHT-pp trajectory with priority intensity map



(c) Non-Homogeneous PMHT-pp trajectory with non-uniformly spatially distributed locales

Figure 4.22: PMHT-pp with Priority Map Comparison

4.12 Conclusion

This chapter has introduced a method of using PMHT to perform path planning to coordinate multiple platforms to explore an environment. The PMHT path planning algorithm coordinates multiple platforms by treating the locations to visit as measurements and the platforms as targets. A novel aspect of the data association problem is that the measurements have no inherent temporal relationship. The PMHT association stage determines probabilities for each pairing of locale and platform at each possible time and then estimates the platform states by taking the expectation over these assignments.

Simulation experiments were used to demonstrate the effectiveness of PMHT for path planning. This method was demonstrated with various platform and locale configurations.

The tradeoff between the smoothness of the tracks and the proximity of the platforms to the locales was found to be controlled by the ratio between the target model noise, Q, and the locale filtering penalty function, R, analogous to measurement noise. As governed by the system and measurement equations, it is expected that as the system noise is reduced, the target manoeuvrability is reduced resulting in smoother trajectories. Similarly, as the "measurement noise" term is reduced, the platforms move closer to the locales.

The scalability of the PMHT path planning problem was investigated both in terms of computation and performance as the number of landmarks was increased. It was found that the computation time increased linearly with the number of landmarks required to be processed. The performance relied on the locations of the locales and the area that was covered by the sensors on the platform. If the locales were packed more densely into the same area for exploration, it was found that the total distance required to cover all the locales was similar and trajectories seemed to converge. In the uniform grid formation, it was found that as the number of locales in the grid was increased but the same area was covered, the trajectories seemed to converge to a similar path.

This chapter compared the performance of the PMHT path planning method with an alternative based on a variant of the travelling salesman problem. The GA-TSP alternative approach was solved using a genetic algorithm. Since the GA-TSP does not incorporate dynamic constraints, the resulting sequences of locales were used as input to the PMHT path planning algorithm to produce smoothed trajectories.

Simulation experiments were used to demonstrate the effectiveness of the path planning algorithms. The methods were demonstrated with various platform and locale configurations.

In terms of total distance travelled by all the platforms, the PMHT-pp performed the best and this was attributed to the ability of the algorithm to efficiently cover regions where multiple locales of interest were close together.

The extension of the batch planning algorithm to a sliding window was discussed. This technique allows for the trajectories of the platforms to be replanned depending on the current states of the platforms. This can be advantageous to correct for localisation errors or to replan for new locales of interest to visit. This method may encourage the platforms to reassociate the locales depending on the position the platforms are in, which may encourage platforms to cross paths.

A particle filter was used as the state estimator in the PMHT-pp to produce platform trajectories to explore an indoor environment. The particle filter made it feasible for non-linear motion and for a non-Gaussian distribution to be estimated, which allowed the trajectories to travel around walls and obstructions. The particle filter also allowed constraints in the target motion to be implemented to control the behaviour of the platform, such as not being able to traverse across walls. The effectiveness of PMHT-pp-pf for indoor path planning was demonstrated with various platform, locale and wall configurations.

Finally, this chapter extended the PMHT-pp method to instead guide the platforms using a non-homogeneous intensity map. This allows for areas of higher importance and removes subjective parameters, such as the spacing of a discrete locale grid. In addition, the intensity guided PMHT-pp produces an intuitive result that the discrete-locale method is simply a numerical approximation to the intensity PMHT-pp.

Chapter 5

PMHT-c for SLAM

5.1 Introduction

Simultaneous localisation and mapping (SLAM) is the problem of creating a map of an unknown area while concurrently estimating the location of the sensing platform within the map. This chapter is concerned with feature-based SLAM, where the map is represented by a collection of landmarks at (unknown) fixed locations. This problem may be difficult because the unknown landmark locations and the unknown platform trajectory are coupled through the (typically non-linear) measurement process. Many SLAM solutions use the statistical assumption that the landmark locations and the platform states are random variables, as introduced by Smith et al. [SSC90]. One method of solution is then to stack the unknowns into a single state vector and employ an Extended Kalman Filter (EKF) [SSC90].

This EKF approach has been widely accepted in the field of robotics as it has been proven to converge to a relative map with zero uncertainty, but it is not without its limitations. The main limitation of EKF based approaches is the computational complexity. The state vector may become very large and hence the covariance matrix of its estimate may be unwieldy. Maintaining the cross-correlation elements of the covariance matrix for K landmarks results in algorithms with computational complexity $O(K^2)$. All the elements must be updated by the Kalman filters with each update, even if just a single landmark has been observed. It can be seen that this problem gets worse as the number of landmarks being observed increases. Another shortcoming is that the EKF may be an inadequate approximation for the non-linear problem.

One solution to this complexity shortcoming is to update only those landmarks that are

within the local area of the platform. The information gained in the local area is eventually incorporated into the global map as a once off cost in computational complexity. Guivant and Nebot [GN01] describes such an algorithm. Although updating the landmarks in the local area of the platform still has quadratic complexity, it no longer has complexity quadratic in the global number of landmarks. This can be a considerable saving if the number of local landmarks is much smaller than the total number of landmarks.

Much subsequent research on SLAM has focused on devising algorithms with a lower computational cost. These approaches include the use of particle filtering [MTKW02], application of the extended information filter [TLK$^+$01] and the use of the Probabilistic Multi-Hypothesis Filter (PMHT) [Dav05]. Thrun [Thr02] gives a good overview of SLAM and some of the approaches to solve the problem. Due to the number of different approaches, only those relevant to the research direction chosen will be described.

An approach using particle filtering was introduced by Montemerlo et al. [MTKW02, MTKW03, MT03] and is referred to as FastSLAM. FastSLAM is based on a Bayesian factorisation of SLAM: when conditioned on a sensor location, the landmark estimation problem is independent for the different landmarks. Due to this, parallel filtering can be used. FastSLAM uses a particle filter to track the sensor location and parallel Extended Kalman Filters to track the landmarks. The pdf of the sensor state is represented by a number of random samples, referred to as particles. For FastSLAM, each particle performs parallel EKFs for each landmark.

As the SLAM sensor platform dynamics may be highly non-linear, the Unscented Kalman Filter (UKF) may also be utilised. As described in chapter 2, the UKF uses the Unscented Transform (UT), a method to calculate the statistics of a random variable that undergoes a nonlinear transformation. The UT is built on the idea that it is easier to approximate a probability distribution than an arbitrary nonlinear function. An Unscented Particle Filter (UPF) and UKF approach to SLAM, which is similar to FastSLAM have been proposed by Wang and Zhang [WZ07]. The UKF can also be used as the estimation scheme in the PMHT, or as a direct approach to solve SLAM.

The Extended Information Filter (EIF) is a version of the Kalman filter used to perform SLAM [MWBDW02, TL05, BS06b]. The EIF is a mathematically equivalent form of the EKF which uses an information matrix and an information vector to represent the estimate rather than the standard mean state and covariance matrix. The information state and matrix are the first and second moments of the log likelihood. The advantage is that the EIF can optimally combine two estimates of a state together by simply adding their information matrices and adding their

information vectors, provided that the errors in each estimate are not correlated to one another. This means that the information fusion of an arbitrary number of platforms becomes a simple sum of the estimates of each platform.

Much of the SLAM literature is based on state estimation methods. As with target tracking, data association may also be important, but this is an issue not usually considered. In most cases, it is explicitly assumed that the association of measurements to landmarks is known [Thr02]. In practice, solutions such as nearest neighbour association are used. It is important that the platform identify and associate landmarks with current observations accurately to minimise errors in both the estimation of localisation and mapping.

The PMHT algorithm has been used to perform data association within SLAM [Dav07]. It was shown to result in state estimation performance superior to various hard assignment alternatives. The paper gave a PMHT formulation for a multi-sensor SLAM, but applied the algorithm to a data set with only one platform. For the estimation scheme, a bank of parallel Kalman filters can be used for linear Gaussian statistics. For the nonlinear data set, an EKF was used.

SLAM is generally performed by a mobile platform that has a variety of different sensors on-board. In addition to landmark feature detection, other information may be available, typically classification measurements from automatic target recognition algorithms that help associate measurements with particular landmarks. Landmark classification and recognition can be used to aid the data association process. This may assist by recognising that a landmark has been revisited in cases where the position measurements are ambiguous. Classification may also provide a means to recognise moving targets and landmarks that can dynamically change. For example, by classifying a particular landmark as having certain properties (such as a door being opened or closed), appropriate models can be applied.

PMHT has been extended to make use of classification information to improve data association when the kinematic information is ambiguous, and this extension is referred to as the PMHT-c [DGS02]. An important trait of PMHT-c is that it allows the use of classification estimates which may be inaccurate, rather than assuming that the classifier has perfect accuracy. This chapter extends the existing PMHT SLAM method of Davey [Dav07] to incorporate classification information and thereby improve data association accuracy. It follows the PMHT-c approach of assuming that the classifier provides imperfect observations of the true landmark class.

This chapter focuses on the use of classification measurements to improve data associa-

tion while performing SLAM. The source of these classification measurements is application dependent and will not be explored here. The chapter assumes that for each of the sensor measurements, there is a classification observation available. The set of classes is known, and the statistical accuracy of the classifier is known through knowledge of the classifier's confusion matrix. As an example, classes may discriminate landmarks based on size or colour. The selection of appropriate features for the classification input is an active field for outdoor environments, e.g. [NCH06, RNDW07], indoor environments, e.g. [FK06], and multi-environments, e.g. [AZA04].

The remainder of the chapter is arranged as follows. Sections 5.2 and 5.3 formulate the SLAM problem. Section 5.4 derives the PMHT-c for SLAM. Section 5.5 demonstrates some of the results achieved on simulated data and on an experimental data set collected in Victoria Park, Sydney, Australia.

5.2 Problem Formulation

In SLAM the problem is to jointly estimate the location of the platform(s) and to estimate a map by which the platform may navigate. This chapter addresses feature based SLAM, where the map consists of a set of discrete landmarks. The SLAM problem is very similar to the multi-target tracking problem, but in this case multiple sensors are assumed and their sensor states and additional classification measurements have to be included. The full problem formulation is presented for completeness.

Assume that there are P sensors observing M landmarks over T observation times, and that not all of these landmarks may be visible to the sensors at any one time. Some (or all) of the sensors may be situated on common platforms.

Let the state of sensor p at scan t be denoted by y_t^p and the set of all states of sensor p over all scans as Y^p. Similarly, let the state of landmark m at scan t be denoted by x_t^m and the set of all states of the mth landmark at all scans by X^m. It is assumed that the dimension of the state vector for each landmark and sensor is fixed, but may vary between landmarks and sensors. It is assumed that redundant states are not estimated if there are multiple sensors situated on the same platform. The sensor state consists of the position, velocity and orientation of the sensor, whereas the landmark state only contains the landmark position since landmarks are stationary.

It is assumed that the prior distribution of the state of each sensor is known and is given by $\phi_0^p(y_0^p)$ for sensor p. The sensor dynamics are also assumed to be known and can be described

by the evolution probability density function (pdf) $\phi_t^p(y_t^p|y_{t-1}^p)$.

Similarly, the prior distribution of each landmark is given by $\psi_0^m(x_0^m)$ for landmark m. In practice, this prior is not known, but it may be estimated from data. As the landmarks are assumed to be stationary, the evolution of the pdf is known.

Let the nth measurement at time t for sensor p be z_{ntp}^x and let the source of the measurement be denoted by k_{ntp}. The assignment variable k_{ntp} is often assumed to be known in SLAM problems, but for this chapter it is unknown and its estimation is the data association process. At time t there are N_{tp} measurements from sensor p. N_{tp} may be zero. Let Z^x denote the set of all measurements and K denote the set of all assignments.

The observation process is described by a known measurement pdf that is denoted by $\zeta^p(z_{ntp}^x|y_t^p, x_t^m, k_{ntp} = m)$. The measurement pdf may be time-varying, but has been assumed constant here to somewhat simplify notation. The particular form of the measurement pdf may be dependent on the sensors used in the application. In the case where the sensor may produce false detections, the pdf of the false detections is denoted by $\zeta^p(z_{ntp}^x|y_t^p, x_t^m, k_{ntp} = 0)$ and may be assumed to be uniform over the observation space in a simple case.

Let there be a known set of classes of landmarks. Each landmark has an associated class, θ^m, which is a static parameter of the landmark and will be assumed known here. In practice, the landmark class must be learned from data, e.g. [RNDW07].

Assume that each measurement has an associated classification measurement, z_{ntp}^k, which is a discrete random variable taken from the known set of landmark classes and an observation of the true landmark class, θ^m. False detections are given a random classification measurement from the known set of landmark classes. Let Z^k denote the set of all classification measurements and Z be the union of Z^x and Z^k, i.e. all of the observed data.

The conditional pmf of the classification measurements, Z^k, is described by a confusion matrix [TK99]. The elements of the confusion matrix are denoted as $c(i,j) \equiv P(z_{ntp}^k = i|\theta^m = j)$ with $C = \{c(i,j)\}$. The confusion matrix is assumed to be independent of the measurement index n and time independent.

5.3 SLAM formulation

This section formulates the standard SLAM problem with sensor measurements only. The standard method of performing SLAM is similar to that of target tracking. Each platform has a state vector which is usually simply its position, heading and speed. Similarly, each landmark

also has a state vector which includes its position. The standard solution is then to stack the state vectors of the various known objects and to use a Kalman Filter to perform the tracking. The first approach to perform SLAM used the EKF due to the nonlinearity [SSC90].

The stacked system vector from the problem definition contains M unknown landmark state vectors and P unknown sensor positions. The stacked vector of all the states at scan t is defined as

$$\Xi_t = [\underbrace{{y_t^1}^\mathsf{T}, \dots {y_t^P}^\mathsf{T}}_{\text{Sensors}}, \underbrace{{x_t^1}^\mathsf{T}, \dots {x_t^M}^\mathsf{T}}_{\text{Landmarks}}]^\mathsf{T}. \tag{5.1}$$

If it is assumed that the assignments are known and that the random processes are linear and Gaussian, the following are known:

$$\phi_0^p(y_0^p) \sim N(\bar{y}_0^p, P_0^{y,p}), \tag{5.2}$$
$$\phi_t^p(y_t^p|y_{t-1}^p) \sim N(F_t^{y,p} y_{t-1}^p, Q_t^{y,p}), \tag{5.3}$$
$$\psi_0^m(x_0^m) \sim N(\bar{x}_0^m, P_0^{x,m}), \tag{5.4}$$
$$\psi_t^m(x_t^m|x_{t-1}^m) \sim N(F_t^{x,m} x_{t-1}^m, Q_t^{x,m}), \tag{5.5}$$
$$\zeta^p(z_{ntp}^x|y_t^p, x_t^m, k_{ntp} = m) \sim N(H_t^{y,p} y_t^p + H_t^{x,m} x_t^m, R_t^p). \tag{5.6}$$

However, the SLAM problem is generally non-linear and the H matrix above is usually replaced, such as by Jacobians if an EKF were to be used.

As the state vector is a stacked vector, the assignments and random processes can be rearranged accordingly. The evolution pdf for the system state is also linear and Gaussian and is given by

$$p(\Xi_t|\Xi_{t-1}) \sim N(F_t \Xi_{t-1}, Q_t), \tag{5.7}$$

where F_t and Q_t are block diagonal matrices, e.g.

$$F_t = \begin{bmatrix} F_t^{y,1} & 0 & & \cdots & & 0 \\ 0 & \ddots & \ddots & & & \\ & \ddots & F_t^{y,P} & 0 & & \vdots \\ \vdots & & 0 & F_t^{x,1} & \ddots & \\ & & & \ddots & \ddots & 0 \\ 0 & & \cdots & & 0 & F_t^{x,M} \end{bmatrix}. \tag{5.8}$$

The mean of the measurement pdf for measurement z^x_{ntp} depends on both the platform and landmark states, which can be written as

$$H^{y,p}_t y^p_t + H^{x,m}_t x^m_t = H_{ntp}\Xi_t, \tag{5.9}$$

where H_{ntp} is a sparse matrix with $H^{y,p}_t$ and $H^{x,m}_t$ positioned appropriately

$$H_{ntp} = [\ldots 0 \ldots H^{y,p}_t \ldots 0 \ldots H^{x,m}_t \ldots 0 \ldots]. \tag{5.10}$$

A stacked measurement matrix can be constructed

$$H_t = \begin{bmatrix} H_{1t1} \\ \vdots \\ H_{N^1_t t1} \\ H_{1t2} \\ \vdots \\ H_{N^P_t tP} \end{bmatrix}, \tag{5.11}$$

and a stacked measurement vector

$$Z^x_t = [{z^x_{1t1}}^\mathsf{T}, \ldots {z^x_{N^1_t t1}}^\mathsf{T}, {z^x_{1t2}}^\mathsf{T}, \ldots {z^x_{N^P_t tP}}^\mathsf{T}]^\mathsf{T}, \tag{5.12}$$

so that the probability of all of the sensor measurements in scan t is

$$p(Z^x_t|y_t, X_t, K_t) = p(Z^x_t|\Xi_t, K_t) \sim N(H_t\Xi_t, R_t), \tag{5.13}$$

where R_t is a block diagonal matrix made up of the covariances corresponding to each individual measurement.

With these rearranged system and measurement expressions, a Kalman filter can be implemented to estimate Ξ_t from the linear Gaussian conditions.

For nonlinear problems, nonlinear extensions to the Kalman Filter can be used such as the EKF and UKF with the stacked state vector. Alternatively, the stacked state vector can be used with nonlinear filters such as the particle filter.

5.4 The PMHT with Classification in SLAM

The derivation of the PMHT SLAM with classification measurements follows the same development as the standard PMHT SLAM given in detail in [Dav07]. The extension of classification to PMHT was introduced by [DGS02] and is repeated here.

In EM terminology, for the SLAM problem, the complete data are (X, Y, K, Z), the incomplete data are (X, Y, Z) and the K are the missing data. The auxiliary function is the expectation of the complete data log-likelihood over the missing data, which now takes the form:

$$Q(X, Y, \Pi|\hat{X}(i), \hat{Y}(i), \hat{\Pi}(i)) = \sum_K P(K|\hat{X}(i), \hat{Y}(i), Z; \hat{\Pi}(i), C) \log P(X, Y, K, Z; \Pi, C), \tag{5.14}$$

where the summation is over all permutations of the assignment variable K. Note that the confusion matrix has implicitly been assumed to be known. If the confusion matrix is unknown, then it may be included as an extra parameter to estimate [DGS02]. However, it has been shown that good performance may be obtained by assuming a fixed confusion matrix, even when there is a large error between the assumed confusion matrix and the truth. [DG01].

Due to the independence assumptions, the complete data likelihood with classification becomes:

$$P(X, Y, K, Z; \Pi, C) = P(X)P(Y)P(K; \Pi)P(Z|X, Y, K; C), \tag{5.15}$$

where

$$P(Y) = \prod_{p=1}^{P} \phi_0^p(y_0^p) \prod_{t=1}^{T} \phi_t^p(y_t^p|y_{t-1}^p), \tag{5.16}$$

$$P(X) = \prod_{m=1}^{M} \psi_0^m(x_0^m) \prod_{t=1}^{T} \psi_t^m(x_t^m|x_{t-1}^m), \tag{5.17}$$

$$P(K; \Pi) = \prod_{n,t,p} \pi_{tp}^{k_{ntp}}, \tag{5.18}$$

and

$$\begin{aligned} & P(Z|X,Y,K;C) \\ &= \prod_{n,t,p} P\left(z^x_{ntp}, z^k_{ntp}|X,Y,K;C\right), \\ &= \prod_{n,t,p} \left\{ p\left(z^x_{ntp}|x_t^{k_{ntp}}, y_t^p\right) p\left(z^k_{ntp}|\theta^{k_{ntp}};C\right)\right\}, \\ &= \prod_{n,t,p} \left\{ \zeta^p(z^x_{ntp}|x_t^{k_{ntp}}, y_t^p)\; c\left(z^k_{ntp}, \theta^{k_{ntp}}\right)\right\}. \end{aligned} \tag{5.19}$$

The conditional probability of the missing data, $P(K|\hat{X}(i), \hat{Y}(i), Z; \hat{\Pi}(i), C)$, can be determined similar to the PMHT derivation in chapter 2, using Bayes' Rule:

$$\begin{aligned} P(K|\hat{X}(i), \hat{Y}(i), Z; \hat{\Pi}(i), C) &= \frac{P(X,Y,K,Z;\Pi,C)}{P(X,Y,Z;\Pi,C)} \\ &= \frac{P(X,Y,K,Z;\Pi,C)}{\sum_K P(X,Y,K,Z;\Pi,C)} \\ &= \frac{P(X)P(Y)P(K;\Pi)P(Z|X,Y,K;C)}{P(X)P(Y)\sum_K P(K;\Pi)P(Z|X,Y,K;C)} \\ &= \frac{P(K;\Pi)P(Z|X,Y,K;C)}{\sum_K P(K;\Pi)P(Z|X,Y,K;C)} \\ &= \frac{\prod_{n,t,p} \pi_{tp}^{k_{ntp}} \zeta^p(z^x_{ntp}|y_t^p, x_t^{k_{ntp}}) c\left(z^k_{ntp}, \theta^{k_{ntp}}\right)}{\sum_K \prod_{n,t,p} \pi_{tp}^{k_{ntp}} \zeta^p(z^x_{ntp}|y_t^p, x_t^{k_{ntp}}) c\left(z^k_{ntp}, \theta^{k_{ntp}}\right)} \\ &= \prod_{n,t,p} \frac{\pi_{tp}^{k_{ntp}} \zeta^p(z^x_{ntp}|y_t^p, x_t^{k_{ntp}}) c\left(z^k_{ntp}, \theta^{k_{ntp}}\right)}{\sum_{m=0}^{M} \pi_{tp}^{m} \zeta^p(z^x_{ntp}|y_t^p, x_t^{m}) c\left(z^k_{ntp}, \theta^{m}\right)} \\ &\equiv \prod_{n,t,p} w_{ntp}^{k_{ntp}} \end{aligned} \tag{5.20}$$

Thus the conditional probability of the assignments is given by the product of individual per measurement *weights*. Each weight, w^m_{ntp}, is the normalised likelihood of the nth measurement

from sensor p at time t being due to landmark m. The difference between these weights and those of the standard PMHT is the inclusion of a confusion matrix term.

Substituting equations (5.15) and (5.20) into (5.14) leads to the auxiliary function to be maximised:

$$\begin{aligned} & Q(X, Y, \Pi | \hat{X}(i), \hat{Y}(i), \hat{\Pi}(i)) \\ &= \log P(X) + \log P(Y) + \sum_{m=0}^{M} \sum_{n,t,p} w_{ntp}^{m} \log \pi_{tp}^{m} \\ &\quad + \sum_{m=0}^{M} \sum_{n,t,p} w_{ntp}^{m} \log \zeta^{p}(z_{ntp}^{x} | y_{t}^{p}, x_{t}^{m}) \\ &\quad + \sum_{m=0}^{M} \sum_{n,t,p} w_{ntp}^{m} \log c\left(z_{ntp}^{k}, \theta^{m}\right) \qquad (5.21) \\ &\equiv Q_{XY} + Q_{\Pi} + Q_{C} \qquad (5.22) \end{aligned}$$

The term Q_C in (5.22) is given by

$$Q_C \equiv \sum_{m=0}^{M} \sum_{n,t,p} w_{ntp}^{m} \log c\left(z_{ntp}^{k}, \theta^{m}\right),$$

and depends only on the classification measurements, the confusion matrix and the landmark classes. These are all known quantities, so Q_C is constant and can be ignored when maximising. If the confusion matrix were unknown, maximising this term would lead to its estimate.

The term Q_Π in (5.22) is identical to that of the standard PMHT as derived in Chapter 2.

The remaining term, Q_{XY}, couples the landmark states, the sensor states and the state measurements and is given by

$$Q_{XY} \equiv \log P(X) + \log P(Y) + \sum_{m=0}^{M} \sum_{n,t,p} w_{ntp}^{m} \log \zeta^{p}(z_{ntp}^{x} | y_{t}^{p}, x_{t}^{m}). \qquad (5.23)$$

For Gaussian measurement noise, it can be shown that this function is equivalent to the log

likelihood of a SLAM problem with known data association [Dav07],

$$Q_{XY} \equiv \log P(X) + \log P(Y) + \sum_{m=0}^{M} \sum_{t=1}^{T} \sum_{p=1}^{P} \log \tilde{\zeta}^p(\tilde{z}_{tp}^m | y_t^p, x_t^m), \tag{5.24}$$

where the synthetic measurement, $\tilde{z}_{tp}^m$, is given by

$$\tilde{z}_{tp}^m = \frac{1}{N_{tp}\hat{\pi}_{tp}^m(i+1)} \sum_{n=1}^{N_{tp}} w_{ntp}^m z_{ntp}^x, \tag{5.25}$$

and the synthetic measurement function, $\tilde{\zeta}_t^p(\cdot)$, is a Gaussian distributed random variable with the same mean as the true measurement function and a variance that is a scaled version of the sensor measurement variance, R,

$$\tilde{R}_{tp}^m = \frac{1}{N_{tp}\hat{\pi}_{tp}^m(i+1)} R. \tag{5.26}$$

Having now arrived at a known-association SLAM problem, any suitable SLAM estimation algorithm may be employed to find the state estimates, using the synthetic measurement and measurement variance as inputs. In the case where the measurement function is Gaussian but nonlinear, the EKF may provide adequate accuracy.

The PMHT-c SLAM consists of iteratively calculating assignment weights, w_{ntp}^m and estimating the sensor and landmark states and assignment priors until convergence. The algorithm is summarised in Algorithm 6. A listing of Matlab source code to calculate the weights for the PMHT-c SLAM algorithm is in Appendix A.3.

5.5 Performance Analysis

The performance of PMHT-c SLAM was investigated through simulation and with real sensor data collected by the University of Sydney.

5.5.1 Simulated results

The effectiveness of the PMHT-c SLAM algorithm described above is now gauged through Monte Carlo simulations and compared with some alternative data association methods. These

Algorithm 6 PMHT-c SLAM algorithm

1: Initialise sensor and landmark state estimates, measurement-to-landmark assignment priors.
2: Calculate the assignment weight for each measurement and landmark track at each scan, using the classification information, according to (5.20).
3: Update the assignment priors using (2.60).
4: Determine synthetic measurements and covariances for each landmark at each scan using (5.25) and (5.26).
5: Update the sensor and landmark state estimates using a Kalman smoothing algorithm over the synthetic measurements and covariances.
6: repeat steps 2 to 5 until convergence.

alternative data association methods, which have been described in chapter 2, include

1. Local Nearest Neighbour (LNN) : A simple association strategy where each landmark within the sensor's range is associated with the closest measurement within a gated distance.

2. Nearest Neighbour - Joint Probabilistic Data Association (NN-JPDA) : An approximate version of JPDA that picks the single best joint event between a target and measurement.

3. PMHT : The version of the PMHT SLAM without classification is used so that direct comparisons with the improvement that classification provides can be seen. Two versions of the PMHT SLAM has been used in the comparison. The first using an EKF estimator and the second using an UKF estimator. For clarity, the EKF version will be labeled as the "PMHT EKF" and the UKF version will be labelled the "PMHT UKF"

4. PMHT-c : Similarly, there were two versions of the PMHT SLAM with classification used in the comparison. The first using an EKF estimator and the second using an UKF estimator. For clarity, the EKF version will be labeled as the "PMHT-c EKF" and the UKF version will be labelled the "PMHT-c UKF".

The simulation scenario consisted of a single sensor platform moving through a field of randomly placed landmarks. To compare the data association effectiveness, the state vector is initialised with the landmark positions. The platform's speed and turn rate have Gaussian process noise with covariance Q, and the position and orientation were obtained by integrating them. The sensor observed landmarks with a field of view limited to 80m in range and $\pm(\pi/2)$

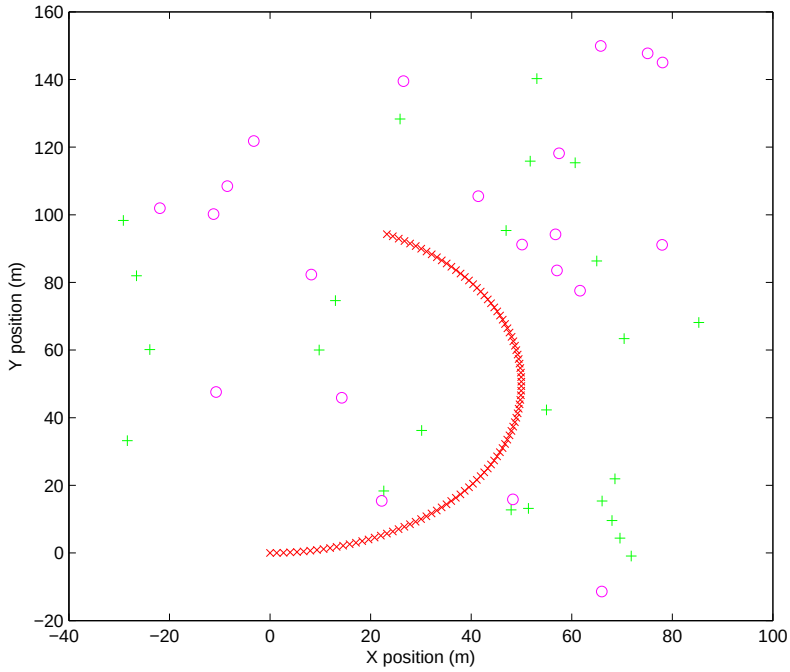


Figure 5.1: Example platform trajectory with random landmarks.

radians in angle. Figure 5.1 shows a typical realisation of the platform motion and the randomly generated landmarks around the platform. In the simulation scenario, the landmarks were randomly classified into two types. The circles and pluses in figure 5.1 represent the landmark class. 200 scans were simulated and statistics were averaged over 100 Monte Carlo realisations.

The classification measurement used for the PMHT-c SLAM simulations had a confusion matrix of the form

$$C = \begin{bmatrix} \alpha & 1-\alpha \\ 1-\alpha & \alpha \end{bmatrix}, \tag{5.27}$$

with $\alpha = 0.9$. It is noted that the rows of the confusion matrix do not need to sum to one and that α does not have to be used in both columns. In this particular simulation, with the two types of landmarks considered, this confusion matrix is appropriate.

The effectiveness of data association was quantified with two estimation accuracy metrics. The first was the percentage of divergent trials. Here, a trial is declared divergent if the instantaneous platform position error was more than 3m at any time during the trial. This indicates

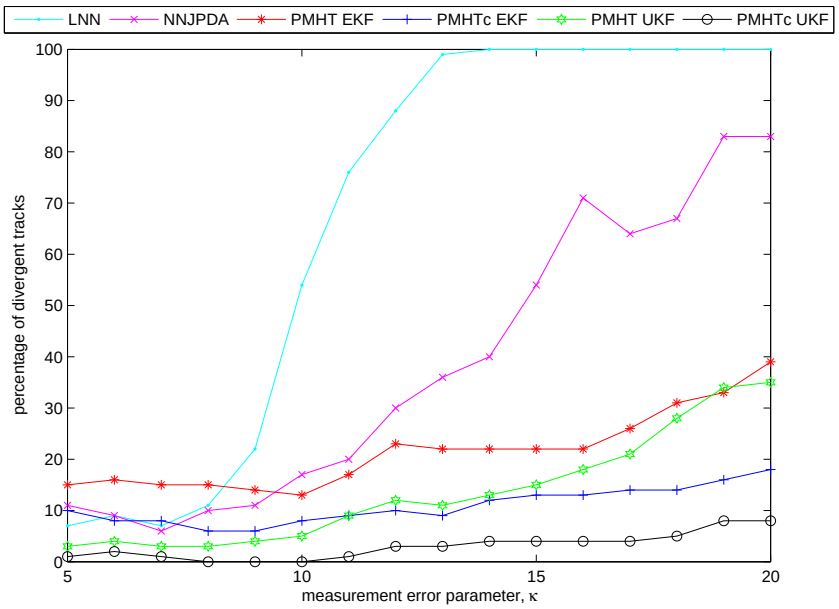


Figure 5.2: Percentage of divergent tracks for different data-association schemes.

a failure of the SLAM algorithm to localise the platform. The second metric was the RMS position error averaged over time and over the 100 Monte Carlo trials. Divergent trials were not used when calculating this metric.

The metrics were calculated for varying sensor accuracy, which was progressively degraded to increase data association difficulty. The sensor measurement noise variance was $(0.5\kappa)^2$ m^2 in range, where κ was varied, and $(0.01\kappa)^2$ rad^2 in bearing. The percentage of divergent trials and the RMS estimation accuracy are plotted as a function of the measurement accuracy parameter κ in figures 5.2 and 5.3, respectively.

The results in figures 5.2 and 5.3 demonstrate that the addition of classification information improves the performance of data association in this problem. Figure 5.2 shows that the PMHT-c EKF SLAM gave 5 to 20% fewer divergent tracks as the measurement accuracy was decreased when compared with the PMHT EKF SLAM, which was considerably better than the LNN and NN-JPDA association techniques. PMHT-c UKF SLAM gave further improvement compared with the PMHT-c EKF SLAM. Figure 5.3 also shows slight improvement in the RMS error of the platform trajectories.

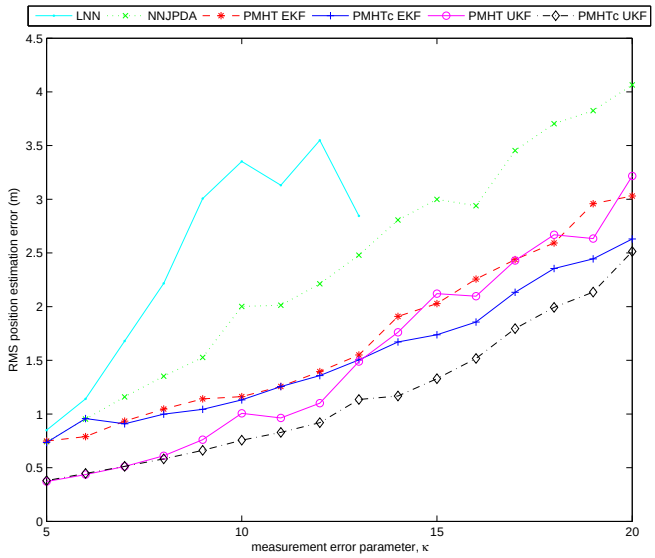


Figure 5.3: RMS position estimation error for different data-association schemes.

Figures 5.4 and 5.5 show a less cluttered view of the results for the PMHT SLAM based algorithms, i.e. 5.2 and 5.3 are repeated showing only the PMHT-based algorithms. In terms of number of divergent tracks and RMS position error, the versions with classification consistently gave better performance compared to the versions without. The PMHT-c UKF SLAM gave approximately half the number of divergent tracks in comparison to the PMHT-c EKF SLAM and slight improvement in RMS error of the platform trajectories.

Figure 5.6 shows the percentage of divergent tracks for PMHT-c EKF as the accuracy of the simulated classification tags was reduced. As would be expected, the number of divergent tracks increases as the number of incorrect classification tags increase. The performance is still quite good, even for very inaccurate classifications, especially if compared with the standard PMHT EKF algorithm (namely the PMHT-c EKF with $\alpha = 0.5$). Note that [DG01] showed that the PMHT-c is equivalent to the standard PMHT for a uniform confusion matrix, ie $\alpha = 0.5$, which was also found to be the case here.

Figure 5.7 illustrates the percentage of divergent tracks for PMHT-c EKF as the accuracy of the simulated classification tags was reduced for a mismatched set value of α. The PMHT-c EKF algorithm assumed a fixed $\alpha = 0.9$, which was thus mismatched to the true classification

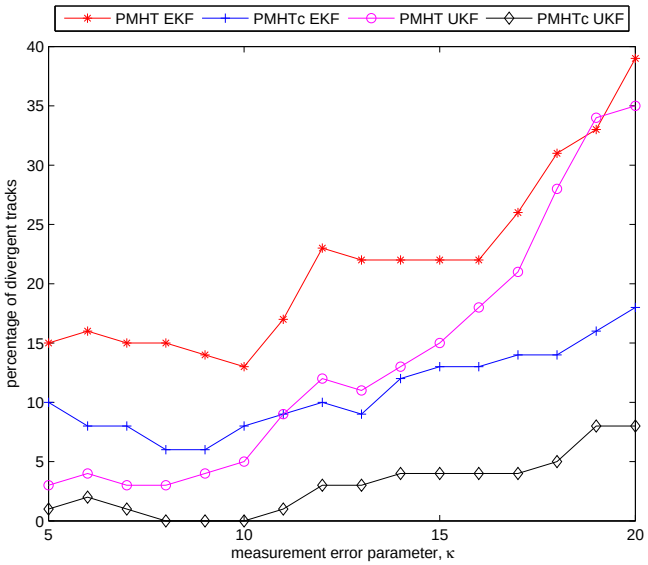


Figure 5.4: Percentage of divergent tracks for different PMHT data-association schemes.

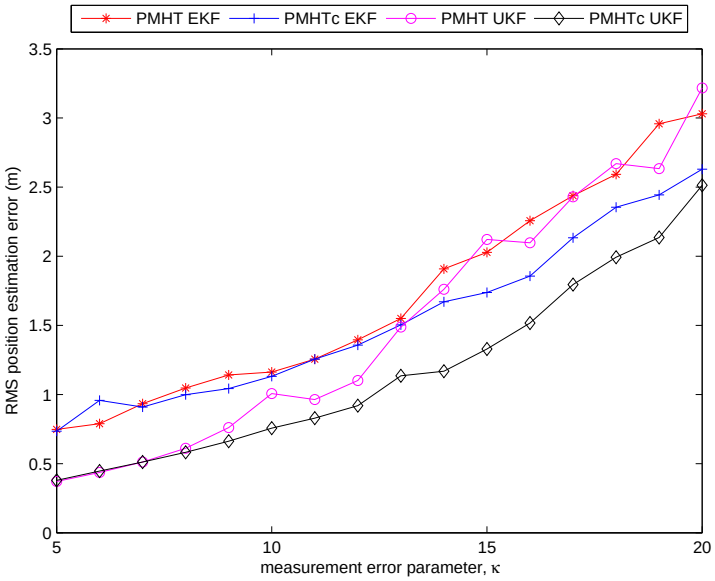


Figure 5.5: RMS position estimation error for different PMHT data-association schemes.

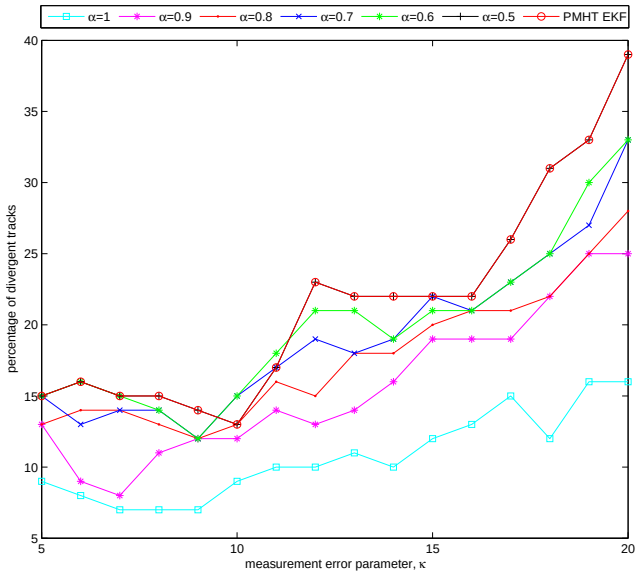


Figure 5.6: Percentage of divergent tracks for different misclassified measurements

measurement accuracy. As expected, the performance degraded as the probability of incorrect classification tags increased. However, the change was minor up to error probabilities of as much as 30% ($\alpha = 0.7$). Even with very poor quality classification information, with only 60% probability of correct classification ($\alpha = 0.6$), the PMHT-c EKF improved performance over PMHT EKF with no classification input.

5.5.2 Victoria Park Data

The Victoria Park data set is a benchmark data set recorded by the University of Sydney. In the experiment, a utility vehicle was fitted with various sensors and driven around Victoria Park (at the University of Sydney), which contains sparsely distributed trees. The sensors onboard included a laser range finder which was used to observe the trees, and inertial sensors that provided measurements of the vehicle's speed and steering direction. GPS data were collected to provide ground truth. Due to occlusion, the GPS signal was not available over the whole experiment. Details about this experiment can be found in [GN01].

Victoria Park was modelled as a 2-D world and the problem was to map out the tree land-

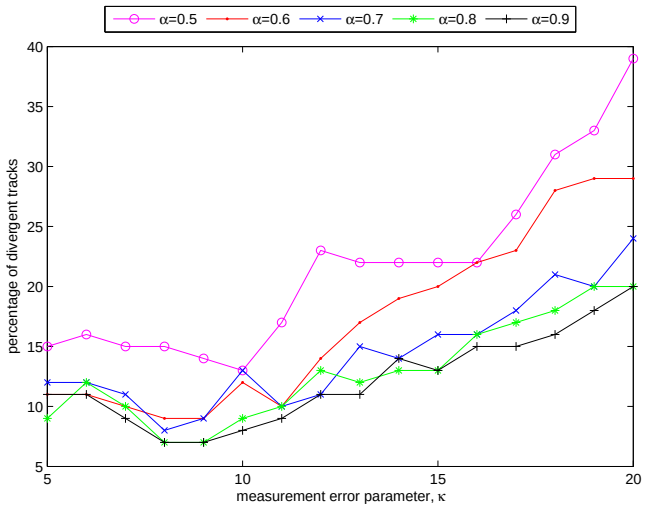


Figure 5.7: Percentage of divergent tracks with mismatched misclassified measurements

marks and to estimate the location of the platform as it moved through this 2-D world. The landmarks were modelled as stationary objects using a simple 2-D position state vector. The single moving platform had a state vector consisting of its 2-D position and its motion, which was approximated using a constant speed and constant turn rate model as in Section 2.2.2,

$$y_t \equiv \left\{ \begin{array}{c} \text{X position} \\ \text{Y position} \\ \text{heading} \\ \text{speed} \\ \text{steering angle} \end{array} \right\}_t \tag{5.28}$$

with white Gaussian process noise on the speed and steering angle to account for manoeuvres. Each sensor was assumed to have white Gaussian measurement noise.

The SLAM algorithm was initialised with only the platform's state vector with the platform's location set at origin. New landmarks were added to the stacked state vector as they were initiated using an ad-hoc landmark initialisation algorithm similar to appendix II in [DNC+01].

The measurements from the laser range finder included the range and bearing of the trees and also gave an estimate of their widths. These widths were placed into a histogram, resulting in the

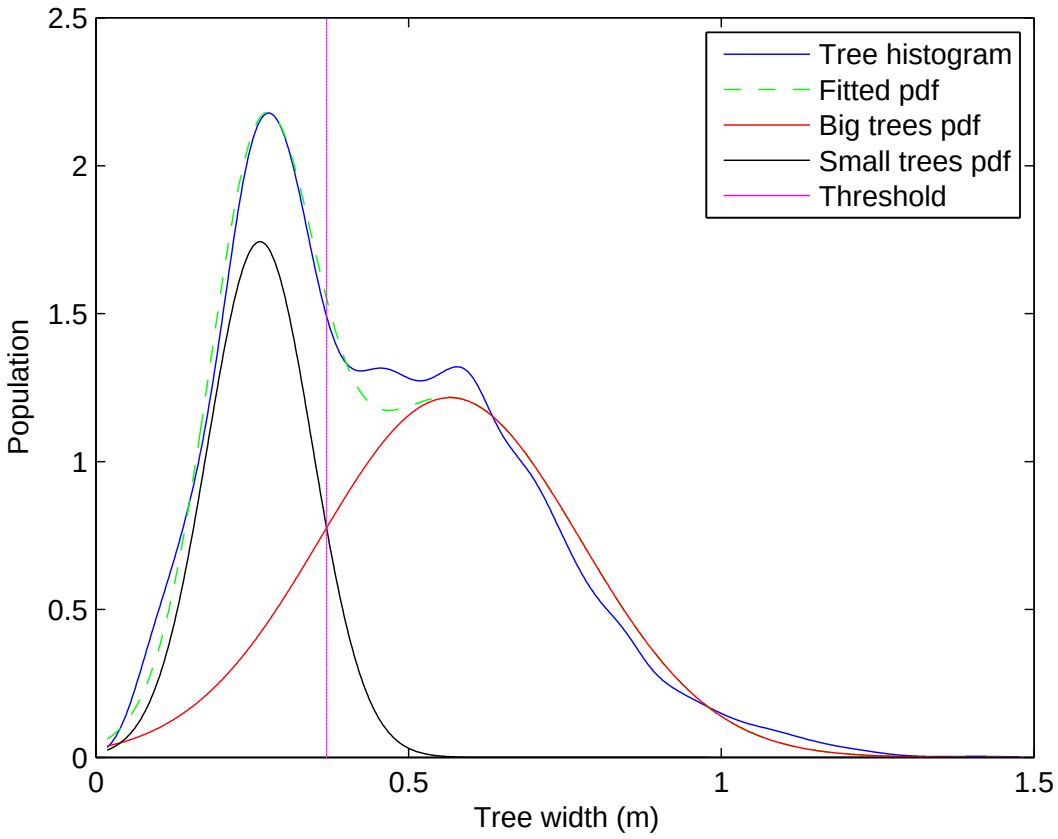


Figure 5.8: Histogram of tree widths

distribution shown in Figure 5.8. The histogram supports the modelling of the tree population as a mixture of narrow and wide trees. Expectation Maximisation was used to fit a mixture of two Gaussian pdfs to the histogram. These two components are also shown in the figure.

A simple classifier was designed to classify the tree landmarks as either “small” or “big” trees by applying a threshold to the observed trunk width. The threshold value was the point where the two fitted Gaussian components intersect, as shown in Figure 5.8.

The PMHT-c EKF SLAM algorithm was run using the laser data only and with the associated classification data based on the above threshold. The platform trajectory estimate is shown in Figure 5.9. The solid line marks the trajectory of the PMHT-c EKF SLAM algorithm and the GPS reports are marked with dots. It is evident that there is error in the GPS measurements

due to fluctuations in the GPS reports, sometimes with jumps of several metres due to satellite occlusions.

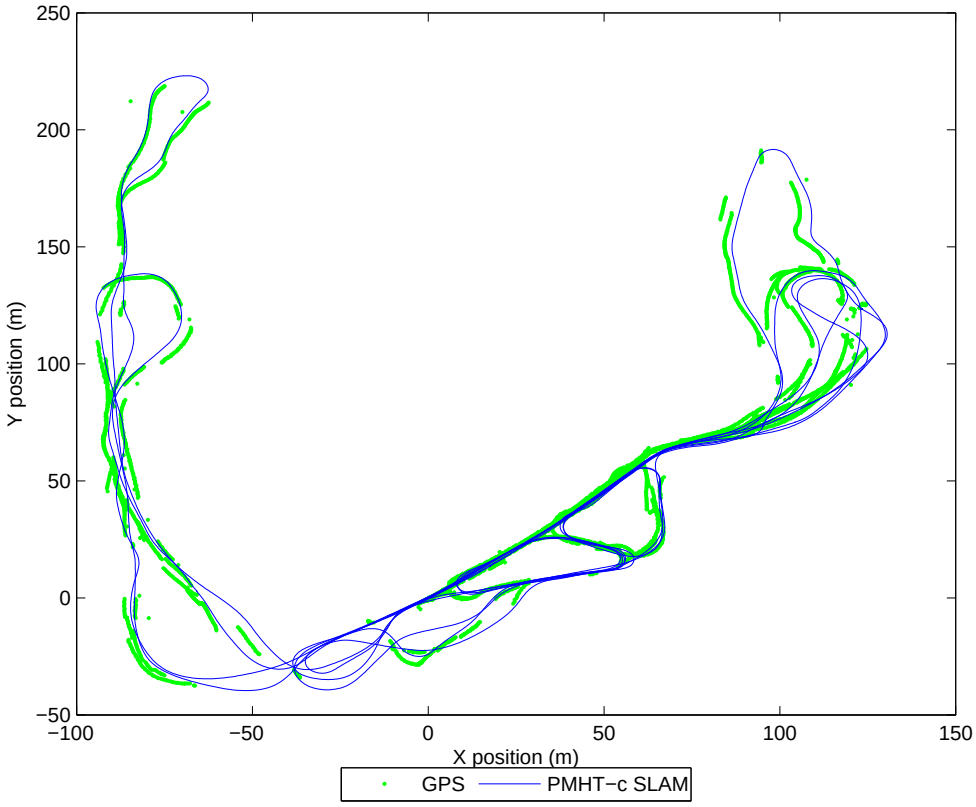


Figure 5.9: PMHT-c estimated trajectory.

The RMS error between the GPS reports and the platform estimate generated using the laser data is approximately 4.5 m for PMHT EKF SLAM and 4.39 m for PMHT-c EKF SLAM. This is within the nominal accuracy of the GPS receiver.

The accuracy of the associated trees were unmeasurable as the true location of the trees were not available in the data set. For a more comprehensive analysis, more experimental testing of this approach is required.

5.6 Conclusions

This chapter used the classification extension to PMHT to improve upon conventional PMHT when solving the SLAM data association problem.

Simulation experiments were used to demonstrate the effectiveness of PMHT-c for data association. Both an EKF and a UKF estimator were used in the comparison. Improvements were seen when compared with PMHT without classification and more common, but simpler data association algorithms, such as nearest neighbour. It was also found that the UKF estimator provided less diverged tracks and lower RMS position errors compared to the EKF estimator.

The Victoria Park data was used to demonstrate the use of PMHT-c on real data with classifications estimated using laser data. The PMHT-c was found to give marginal improvement in the position error.

The use of classification measurements can be very beneficial for SLAM as better data association leads to more accurate landmark estimates and localisation of the platform.

Chapter 6

Conclusion

This chapter summarises each of the contributions of this thesis and concludes with future work.

6.1 Tracking with Time Uncertainty

Relatively little work has been done in the area of estimation with uncertain timing information. The problem of tracking using measurements with time errors needs to be accounted for.

An extension to the PMHT was introduced to perform target tracking for the situation where the temporal information is noisy or unreliable. The key idea was to use the PMHT to associate measurements to time instants as well as to targets by treating both assignments as missing data.

The PMHT-t association algorithm determines probabilities for each pairing of measurement and target at each possible time and then estimates the target states by taking the expectation over these assignments.

Simulations were used to demonstrate the improved performance of the PMHT-t algorithm over the standard PMHT in dealing with measurements with a stochastic time delay.

Further simulations considered a scenario where true measurement times were available occasionally while noisy measurement times were available for all measurements. The time error parameters and target states were estimated simultaneously by the PMHT. The PMHT-t was compared with existing state estimation algorithms for this problem; the Kalman filter and the particle filter. The PMHT-t gave improvements over the best alternative, the particle filter with 500 samples.

6.2 Multiple Platform Path Planning

There have been many strategies devised to control multiple platforms to cooperatively explore an area. The most common approach is to enumerate a collection of hypotheses based on the platforms sensor detections or motion, and use a cost function as the decision criterion.

A method of using PMHT to perform path planning to coordinate multiple platforms to explore an environment was introduced. The PMHT path planning algorithm coordinates multiple platforms by treating the locations to visit as measurements and the platforms as targets. A novel aspect of the data association problem is that the measurements have no inherent temporal relationship. The PMHT-pp association stage determines probabilities for each pairing of locale and platform at each possible time and then estimates the platform states by taking the expectation over these assignments.

Simulation experiments were used to demonstrate the effectiveness of PMHT-pp for path planning. This method was demonstrated with various platform and locale configurations.

The tradeoff between the smoothness of the tracks and the proximity of the platforms to the locales was found to be controlled by the ratio between the target model noise and the locale filtering penalty function, analogous to measurement noise. As governed by the system and measurement equations, it is expected that as the system noise is reduced, the target dynamics is reduced resulting in smoother trajectories. Similarly, as the measurement noise is reduced, the platforms move closer to the locales.

The scalability of the PMHT-pp path planning problem was investigated both in terms of computation and performance as the number of landmarks was increased. It was found that the computation time increases linearly with the number of landmarks required to be processed. The performance relied on the locations of the locales and the area that was covered by the sensors on the platform. If the locales were packed more densely into the same area for exploration, it was found that the total distance required to cover all the locales was not strongly influenced by the locale density.

The performance of the PMHT-pp method was compared with an alternative based on a variant of the travelling salesman problem. The GA-TSP alternative approach was solved using a genetic algorithm. Since the GA-TSP did not incorporate dynamic constraints, the resulting sequences of locales were used as input to the PMHT-pp algorithm to produce smoothed trajectories.

More simulation experiments were used to compare the effectiveness of the various path

planning algorithms. The methods were once again demonstrated with various platform and locale configurations. In terms of total distance travelled by all the platforms, the PMHT-pp performed the best which was attributed to the ability of the algorithm to efficiently cover regions where multiple locales of interest were close together.

The extension of the batch planning algorithm to a sliding window was then discussed. This technique allows for the trajectories of the platforms to be replanned depending on the current states of the platforms. This can be advantageous to correct for localisation errors or to handle changes locale configurations. This method allows the platforms to reassociate the locales to different platforms as they move, which then provides the potential for the platform paths to cross. The crossing of paths has been shown to improve localisation errors by the SLAM community, known as “closing the loop”.

A particle filter was used as the state estimator in the PMHT-pp to produce platform trajectories when there are hard constraints on the motion of the platform, such as an indoor environment. The particle filter made it feasible for non-linear motion and for a non-Gaussian distribution to be estimated, which allowed the trajectories to travel around walls and obstructions. The particle filter also allowed constraints in the target motion to be implemented to control the behaviour of the platform, such as not being able to traverse across walls. The effectiveness of PMHT-pp-pf for indoor path planning was demonstrated with various platform, locale and wall configurations.

Finally the PMHT-pp method was extended to instead guide the platforms using a non-homogeneous intensity map. This allows for areas of higher importance and removes subjective parameters, such as the spacing of a discrete locale grid. In addition, the intensity guided PMHT-pp produces an intuitive result that the discrete-locale method is simply a numerical approximation to the intensity PMHT-pp.

Interestingly, the PMHT was developed for data association in target tracking, which can be considered to be a point-to-point assignment problem. In contrast, by removing the timing information and taking the limiting case of a point-process measurement model, the path planning problem considered in this chapter effectively uses PMHT to assign a curve to a surface.

6.3 SLAM with classifications

SLAM is the problem of creating a map of an unknown area while concurrently estimating the location of the sensing platform within the map. Dependent on the sensors onboard the plat-

form, in addition to landmark feature detection and position information, classification information may be available. Measurements from automatic target recognition algorithms provide classification data that can help associate measurements with particular landmarks.

The classification extension to PMHT was used to improve upon conventional PMHT when solving the SLAM data association problem.

Simulation experiments were used to demonstrate the effectiveness of PMHT-c for data association. Both an EKF and an UKF estimator were used in the comparison. Improvements were seen when compared with PMHT without classification and more common, but simpler data association algorithms, such as nearest neighbour. It was also found that the UKF estimator provided less divergent tracks and lower RMS position errors compared to the EKF estimator.

The Victoria Park data was used to demonstrate the use of PMHT-c on real data with classifications estimated using laser data. The PMHT-c was found to give marginal improvement in the position error.

6.4 Future Work

This thesis introduced new methods to solve several different problems in innovative ways. Most of the work was evaluated via simulations, which leaves much scope for development in a real world application with real data.

The novel method to explore an area with the PMHT-pp leaves many developments for future work. In the example of the sliding batch version of the PMHT-pp, each of the platforms could create an individual map. Further work could investigate methods for connecting the isolated sub-maps to form an overall map of the whole area.

The novel non-homogeneous PMHT-pp algorithm leaves much room for future work. Although the algorithm was demonstrated in a path planning context, it is intuitive to see that the method has application in a wide range of problems, Fundamentally, the PMHT-pp algorithm efficiently associates a finite resource between multiple consumers over a long time horizon with dynamic constraints. It could be applied to many resource management problems. The time evolving intensity map could be used as a means to provide a sensor scheduling algorithm by using a Probability Hypothesis Density [Mah03] map of probable target locations.

Another area of interest is integrating path planning with information theoretics to perform exploration whilst minimising the platform localisation error. This could be combined with the integration between the PMHT-pp and the PMHT-c SLAM algorithm for active exploration.

Appendix A

Source Code Listing

This appendix presents the source code for the major functions used in this thesis.

A.1 PMHT-t Source Code

Chapter 3 introduced an extension to PMHT to track targets using measurements with time stamp errors. The code listing A.1 shows the function to track multiple targets given batches of measurements with time delay errors.

Listing A.1: PMHT-t code

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function generates tracks on measurements with time
% errors using the PMHT-t.
%
% Input parameters:
% xhat - State means represented by 4xTxM matrix. Each column
%        composed of the constant velocity state matrix as
%        described in Section 2.2.2.
% meas - Measurements in the batch to be processed, in a 3xNxT
%        matrix. Each column includes the positional information
%        and the noisy time stamp.
% time_start - First time of batch.
% time_end - Final time of batch
% display_flag - Flag for figures. 1 for on, 0 for off.
%
% Output parameters:
% xhat
%
% Adjustable parameters include the $F$, $P$, $Q$,
% $R$ and $H$ filter matrices and convergence conditions.
```

```
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function [xhat] = runPMHT_t(xhat,meas,time_start,time_end,display_flag)

% Calculate batch size, number of scans, targets and measurements
batch_size = time_end - time_start+1;
num_scans = size(xhat,2);
num_targets = size(xhat,3);
num_meas = size(meas,2);

% Set F, P, Q, R and H matrices
F = [1 1; 0 1];
F = [F zeros(2); zeros(2) F];
P = zeros(4,4,num_scans,num_targets);
Q = eye(4)*1.0e-8;
Rb = eye(2)*1.0e-8;
H = [1 0 0 0; 0 0 1 0];

% Set converged flag and iteration count to 1
not_converged = true;
it_count = 1;

% Set prior values for targets and time
m_prior = ones(num_meas, num_targets) / num_targets;
t_prior = ones(num_meas, num_scans)   / num_scans;

while not_converged
    oldx = xhat;

    R = Rb * (100/it_count)^2;

    % Initialise weights and set clutter weights
    w = zeros(num_meas, num_scans, num_targets+1);
    w(:,:,num_targets+1) = 1/1000;

    % Calculate weights
    for m = 1: num_meas
        for tk = 1: num_targets
            p = 0.7; % Probability of no delay set to 0.7 in this example
            time_noise = zeros(1,batch_size);
            time_noise(1:6) = ([p p*(1-p) p*(1-p)^2 p*(1-p)^3 ...
               p*(1-p)^4 p*(1-p)^5]);
            meas_time = meas(3,m)-time_start+1;
            time_factor = zeros(1,batch_size);
            time_factor(1:meas_time) = fliplr(time_noise(1:meas_time));
            time_factor = time_factor/sum(time_factor);

            w(m, :, tk) = m_prior(m,tk) * t_prior(m,:) .* time_factor ...
```

```
                .* exp(-0.5 * sum((xhat([1 3],:,tk) - meas(1:2, ...
                m)*ones(1, num_scans)).^2));
        end
        nfac = (sum(sum(w(m, :, :))));
        w(m, :, :) = w(m, :, :) / nfac;

        temp_m_prior = squeeze(sum(w(m,:,:),2));
        m_prior(m,:) = temp_m_prior(1:num_targets);
        t_prior(m,:) = squeeze(sum(w(m,:,1:num_targets),3));
        new_t_prior(m) = (time_start-1) + sum(t_prior(m,:).*[1:10]);
    end

    for tk = 1: num_targets
        % Forward recursion
        for t = 2: num_scans
            xhat(:, t, tk) = F * xhat(:, t-1, tk);
            Phat = F * P(:,:,t-1, tk) * F' + Q;
            wsum = sum(w(:, t, tk));
            if wsum > 0.00001
                z = (meas(1:2,:) * w(:, t, tk)) ./ wsum;
                Rs = R / wsum;
                S = H * Phat * H' + Rs;
                W = Phat * H' * inv(S);
                xhat(:, t, tk) = xhat(:, t, tk) + W * (z - H * ...
                    xhat(:, t, tk));
                Phat = Phat - W * S * W';
            end
            P(:,:,t, tk) = Phat;
        end

%       Backward recursion
        for t = num_scans-1: -1 :1
            Ppred = F * P(:,:,t, tk) * F' + Q;
            gain = P(:,:,t, tk) * F' * inv(Ppred);
            xhat(:,t, tk) = xhat(:,t, tk) + gain * (xhat(:,t+1, tk) - ...
                F * xhat(:,t, tk));
        end
    end

%   Plot current iteration of tracks
    if display_flag == 1
        set(0,'CurrentFigure',1)
        hold off
        plot(meas(1,:), meas(2,:), 'cd')
        hold on
        col='bkrg';
        for tk = 1: num_targets
            plot(xhat(1, :, tk),xhat(3, :, tk), [col(tk) '.-'])
        end
```

```
115         xlabel('X position')
116         ylabel('Y position')
117         axis([0 11 0 11])
118         pause(0.1)
119     end
120 
121     % Set convergence flags
122     it_count = it_count + 1;
123     if it_count > 200
124         not_converged = false;
125     end
126     del = sum((oldx(:) - xhat(:)).^2);
127     if del < 0.0005
128         not_converged = false;
129     end
130 end
```

A.2 PMHT-pp Source Code

Chapter 4 introduced a method to use PMHT to perform path planning to coordinate multiple platforms to explore an environment. The example results from Section 4.5 were produced by code listing A.2, which shows the function to generate trajectories for multiple platforms to visit various predefined set of locales of interest using the PMHT-pp.

Listing A.2: PMHT-pp

```
1  %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
2  %
3  % This function generates trajectories for multiple platforms
4  % to visit a predefined set of locales of interest using the
5  % PMHT-pp.
6  %
7  % Presets: 4 platforms initialised at center,
8  %          500 time scans,
9  %          10x10 grid of locales
10 %
11 % The various examples are presented as commented code below
12 % and can be changed accordingly.
13 %
14 %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
15 
16 function outputs = runPathPlanner
17 
18 % Set number of scans, targets and measurements
19 num_scans = 500;
```

```
num_targets = 4;
num_meas = length(meas);

% Create locales of interest

% Generate random locales
% a = rand(1, 100) * 9 + 1;
% b = rand(1, 100) * 9 + 1;

% Generate grid of locales
[a, b] = ind2sub([10,10], (1:100));

% Assign locales as measurements
meas = [a; b];

figure(1),clf
hold off

% Plot locales
plot(meas(1,:), meas(2,:), 'rd')
hold on

% Create 6 element platform state
xhat = zeros(6, num_scans, num_targets);

% Uncomment relevant initial platform positions - these were the ...
    initialisations used in the various scenarios in the thesis.

% Random initial platform positions
%   for plat_index = 1:num_targets
%       xhat(1, :, plat_index) = rand * 9 + 1;
%       xhat(4, :, plat_index) = rand * 9 + 1;
%   end

% 3 platforms for random locales scenario
% xhat(1, :, 1) = 0;
% xhat(4, :, 1) = 4;
%
% xhat(1, :, 2) = 0;
% xhat(4, :, 2) = 5;
%
% xhat(1, :, 3) = 0;
% xhat(4, :, 3) = 6;

% 3 platforms on left
% xhat(1, :, 1) = 0;
% xhat(4, :, 1) = 5;
%
% xhat(1, :, 2) = 0;
```

```
% xhat(4, :, 2) = 5.5;
%
% xhat(1, :, 3) = 0;
% xhat(4, :, 3) = 6;

% 4 Platforms outside
% xhat(1, :, 1) = 0;
% xhat(4, :, 1) = 0;
%
% xhat(1, :, 2) = 0;
% xhat(4, :, 2) = 11;
%
% xhat(1, :, 3) = 11;
% xhat(4, :, 3) = 11;
%
% xhat(1, :, 4) = 11;
% xhat(4, :, 4) = 0;

% 4 platforms inside
xhat(1, :, 1) = 5;
xhat(4, :, 1) = 6;

xhat(1, :, 2) = 6;
xhat(4, :, 2) = 6;

xhat(1, :, 3) = 5;
xhat(4, :, 3) = 5;

xhat(1, :, 4) = 6;
xhat(4, :, 4) = 5;

% Set colours and plot initial positions
col = hsv(num_targets);
for tk = 1: num_targets
    plot(xhat(1, :, tk),xhat(4, :, tk),'s', 'color',col(tk,:))
end
xlabel('X position')
ylabel('Y position')
axis([0 11 0 11])

% Set F, Q, R and H matrices
F = [1 1 0.5; 0 1 1; 0 0 1];
F = [F zeros(3); zeros(3) F];
Q = eye(6)*1.0e-8;
Rb = eye(2)*1.0e-4;
H = [1 0 0 0 0 0; 0 0 0 1 0 0];

% Set initial covariance and weights
P = zeros(6,6,num_scans,num_targets);
```

```
117 m_prior = ones(num_meas, num_targets) / num_targets;
118 t_prior = ones(num_meas, num_scans)   / num_scans;
119
120 % Initialise convergence flag and EM iteration counter
121 not_converged = true;
122 it_count = 1;
123
124 while not_converged
125     oldx = xhat;
126
127     R = Rb * (100/it_count)^2;
128
129     % Calculate weights
130     w = zeros(num_meas, num_scans, num_targets);
131     for m = 1: num_meas
132         for tk = 1: num_targets
133             w(m, :, tk) = m_prior(m,tk) * t_prior(m,:) .* exp(-0.5 * ...
                    sum((xhat([1 4],:,tk) - meas(:, m)*ones(1, ...
                    num_scans)).^2));
134         end
135
136         nfac = (sum(sum(w(m, :, :))) + 1e-4);
137         w(m, :, :) = w(m, :, :) / nfac;
138
139         if nfac>0.0001
140             m_prior(m,:) = squeeze(sum(w(m,:,:),2));
141             t_prior(m,:) = squeeze(sum(w(m,:,:),3));
142         else
143             m_prior = ones(num_meas, num_targets) / num_targets;
144             t_prior = ones(num_meas, num_scans)   / num_scans;
145         end
146     end
147
148     for tk = 1: num_targets
149
150         % Forward recursion
151         for t = 2: num_scans
152             xhat(:, t, tk) = F * xhat(:, t-1, tk);
153             Phat = F * P(:,:,t-1, tk) * F' + Q;
154
155             wsum = sum(w(:, t, tk));
156             if wsum > 0.00001
157                 z = (meas * w(:, t, tk)) ./ wsum;
158                 Rs = R / wsum;
159                 S = H * Phat * H' + Rs;
160                 W = Phat * H' * inv(S);
161                 xhat(:, t, tk) = xhat(:, t, tk) + W * (z - H * ...
                        xhat(:, t, tk));
162                 Phat = Phat - W * S * W';
```

```
            end
            P(:,:,t, tk) = Phat;
        end

        % Backward recursion
        for t = num_scans-1: -1 :1
            Ppred = F * P(:,:,t, tk) * F' + Q;
            gain = P(:,:,t, tk) * F' * inv(Ppred);
            xhat(:,t, tk) = xhat(:,t, tk) + gain * (xhat(:,t+1, tk) - ...
                F * xhat(:,t, tk));
        end
    end

    % Plot results
    set(0,'CurrentFigure',1)
    hold off
    plot(meas(1,:), meas(2,:), 'rx')
    hold on
    for tk = 1: num_targets
        plot(xhat(1, :, tk),xhat(4, :, tk), '.-', 'color',col(tk,:))
    end
    xlabel('X position')
    ylabel('Y position')

    pause(0.1)

    % Set flag if number of iterations reached
    it_count = it_count + 1;
    if it_count > 100
        not_converged = false;
    end
    % Set flag if results have converged
    del = sum((oldx(:) - xhat(:)).^2);
    if del < 0.0005
        not_converged = false;
    end
end

% Uncomment if weight distributions plots required
% figure(2),clf
% imagesc(t_prior)
%
% figure(3),clf
% imagesc(m_prior)

outputs = xhat;
```

A.3 PMHT-c SLAM Source Code

Chapter 5 used the classification extension to PMHT to process the classification data to improve the data association in SLAM. The code listing A.3 shows the function to associate landmark targets to measurements based on both position and classification.

Listing A.3: PMHT-c SLAM

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% This function associates targets with measurements using
% both positional and classification information using the
% PMHT-c.
%
% Input parameters:
% state - Stacked state vector of the platform and landmarks.
% params - structure containing clutter_pdf value and noise
%          covariance matrix R.
% meas - Sensor measurements to be associated, in a 4xN
%        matrix. Each column includes the positional information
%        and the classification information.
%
% Output parameters:
% assoc_data - Structure containing list of measurements
%              associated to landmark targets.
%
% Adjustable parameters include the classification matrix, C,
% and the gate distance.
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function assoc_data = CalculateWeights (state, params, meas)

% Number of elements in platform state for indexing
statel = 5;

% Number of landmark tracks
num_tracks = (length(state) - statel) / 2;

% Initialise association matrices
assoc_meas = [];
targ = [];
wsum = [];

% Set number of sensor measurements
num_meas = size(meas,2);

% Set classification matrix
C = [0.9 0.1; 0.1 0.9];
```

```
41
42  % Initialise weights matrix and clutter weights
43  weights = zeros (num_tracks + 1, num_meas);
44  weights(num_tracks + 1, :) = params.clutter_pdf;
45
46  if num_meas > 0
47
48      for tr = 1: num_tracks,
49          % Calculate range and bearing of landmarks from platform position
50          temp = state(statel + [1:2] + (tr-1) * 2) - state(1:2);
51          range   = sqrt(temp' * temp);
52          bearing = atan2(temp(2),temp(1)) - state(3);
53
54          while bearing > pi
55              bearing = bearing - 2 * pi;
56          end
57          while bearing < -pi
58              bearing = bearing + 2 * pi;
59          end
60
61          % Calculate distance between landmarks and sensor measurements
62          dist = [range; bearing] * ones(1, num_meas) - meas(1:2,:);
63          dd = dist(1,:).^2 / R(1,1) + dist(2,:).^2 / R(2,2);
64
65          % Find measurements within gated distance of 100
66          inds = find(dd < 100);
67
68          % Calculate PMHT-c weights
69          if length(inds) > 0
70              for ind_index = 1:length(inds)
71                  detS = 2*pi*sqrt(Rl(1,1)*Rl(2,2));
72                  weights(tr,inds(ind_index))=1 ./ detS .* exp (-0.5 * ...
                        dd(inds(ind_index))) .* C(track.class(tr)+1, ...
                        meas(4,inds(ind_index))+1);
73              end
74          end
75      end
76  end
77
78  % Normalise weights matrix
79  weights = weights ./ (ones(num_tracks+1, 1) * (sum (weights, 1)));
80
81  % Assign synthetic measurements to targets
82  for tr = 1: num_tracks
83      z = meas(1:2,:) * weights(tr,:)';
84      ww = sum(weights(tr,:));
85
86      if ww > 1.0e-3
87          assoc_meas = [meas; z./ww];
```

```
88          targ = [targ, tr];
89          wsum = [wsum, ww];
90      end
91  end
92  assoc_data.meas = meas;
93  assoc_data.targ = targ;
94  assoc_data.wsum = wsum;
```

Bibliography

[AMGC02] M. Sanjeev Arulampalam, Simon Maskell, Neil Gordon, and Tim Clapp, *A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking*, IEEE Transactions on Signal Processing **50** (2002), no. 2, 174–188.

[AZA04] D. Asmar, J. Zelek, and S. Abdallah, *SmartSLAM: localization and mapping across multi-environments*, IEEE International Conference on Systems, Man and Cybernetics, 2004.

[BMF+00] W. Burgard, M. Moors, D. Fox, R. Simmons, and S. Thrun, *Collaborative Multi-Robot Exploration*, IEEE International Conference on Robotics and Automation, 2000.

[BP99] S. Blackman and R. Popoli, *Design and analysys of modern tracking systems*, Artech House, Norwood, MA, 1999.

[BS06a] M. Bryson and S. Sukkarieh, *Active Airborne Localisation and Exploration in Unknown Environments using Inertial SLAM*, IEEE Aerospace Conference Proceedings, 2006.

[BS06b] _____, *Cooperative Localisation and Mapping for Multiple UAVs in Unknown Environments*, IEEE Aerospace Conference Proceedings, 2006.

[BS09] _____, *Architectures for cooperative Airborne Simultaneous Localisation and Mapping*, Journal of Intelligent and Robotics Systems, Special Issue on Airborne SLAM, 2009.

[BSF88] Y. Bar-Shalom and T. E. Fortmann, *Tracking and data association*, Academic Press, 1988.

[BSL88] Y. Bar-Shalom and X. R. Li, *Multitarget-multisensor tracking: principles and techniques*, YBS Publishing, Storrs, CT, 1988.

[BST75] Y. Bar-Shalom and E. Tse, *Tracking in a cluttered environment with probabilistic data association*, Automatica 11, 1975.

[CKK96] A. Cassandra, L. Kaebling, and J. Kurien, *Acting under uncertainty: Discrete bayesian models for mobile robot navigation*, IEEE International Conference on Intelligent Robots and Systems, 1996.

[Dav05] S. Davey, *Simultaneous Localisation and Map Building using the Probabilistic Multi-Hypothesis Tracker*, Tech. Report DSTO-TR-1691, DSTO, 2005.

[Dav07] S. J. Davey, *Simultaneous localization and map building using the probabilistic multi-hypothesis tracker*, IEEE Transactions on Robotics **23** (2007), 271–280.

[Dav11] _____, *Histogram pmht with particles*, Proceedings of the 14th International Conference on Information Fusion, 2011.

[DFG01] A. Doucet, N. De Freitas, and N. Gordon, *Sequential Monte Carlo Methods in Practice*, Springer Verlag, 2001.

[DG01] S. J. Davey and D. A. Gray, *Sensitivity of the pmht-c algorithm to classifier model mismatch*, Proceedings of the International Conference on Optimisation Techniques and Applications, December 2001.

[DGS02] S. Davey, D. Gray, and R. Streit, *Tracking, Association, and Classification: A Combined PMHT Approach*, Digital Signal Processing **12** (2002), 372–382.

[DLR77] A.P. Dempster, N.M Laird, and D.B. Rubin, *Maximum likelihood from incomplete data via the EM algorithm*, Journal of the Royal Statistics Society (1977), 1–38.

[DNC+01] M. W. M. G. Dissanayake, P. Newman, S. Clark, H. Durrant-Whyte, and M. Csorba, *A solution to the simultaneous localization and map building (slam) problem*, IEEE Trans. Aerosp. Electron. Syst., June 2001.

[Fit90] R. J. Fitzgerald, *Development of practical pda logic for multitarget tracking by microprocessor*, vol. 1, ch. 1, Automatica 11, Norwood, MA: Artech House, 1990.

[FK06] S. Fazli and L. Kleeman, *Simultaneous landmark classification, localization and map building for an advanced sonar ring*, Cambridge University Press **25** (2006), 283–296.

[FLS99] H. Feder, J. Leonard, and C. Smith, *Adaptive mobile robot navigation and mapping*, International Journal of Robotics Research, 1999.

[FNL02] J. Fenwick, P. Newman, and J. Leonard, *Cooperative concurrent mapping and localization*, Proceedings of the IEEE International Conference on Robotics and Systems, 2002.

[GB01] W. R. Gilks and C. Berzuini, *Following a moving target: Monte carlo inference for dynamic bayesian models*, J. R. Statist. Soc. B **63** (2001), 127–146.

[GKC03] R. Grabowski, P. Khosla, and H. Choset, *Autonomous exploration via regions of interest*, IEEE International Conference on Intelligent Robots and Systems, 2003.

[GN01] J. Guivant and E. Nebot, *Optimization of the simultaneous localization and map building algorithm for real time implementation*, IEEE Transaction of Robotic and Automation, 2001.

[Gol89] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley Publishing Company, Inc, 1989.

[Jaz70] A. H. Jazwinski, *Stochastic processes and filtering theory*, Academic Press, New York, 1970.

[JU04] S. J. Julier and J. K. Uhlmann, *Unscented Filtering and Nonlinear Estimation*, IEEE, vol. 92, Mar. 2004, pp. 401–402.

[Kir07] J. Kirk, *Traveling salesman problem - genetic algorithm*, http://www.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm, 2007.

[LJ00] X. R. Li and V. P. Jilkov, *A survey of maneuvering target tracking: Dynamic models.*, Proceedings of the SPIE Conference on Signal and Data Processing of Small Targets, vol. 4048, December 2000, pp. 212–235.

[LL06] Z Li and H Leung, *An expectation maximisation based simultaneous registration and fusion algorithm for radar networks*, IEEE Canadian Conference on Electrical and Computer Engineering (Ottawa, Canada), May 2006, pp. 31–35.

[Mah03] R. Mahler, *Multi-target Bayes filtering via first-order multi-target moments*, IEEE Trans. on Aerospace and Electronic Systems **39** (2003), no. 4, 1152–1178.

[MK97] G. J. McLachlan and T. Krishnan, *The em algorithm and extensions*, Wiley Interscience, 1997.

[Mor08] M. Morelande, *Linear filtering with timing uncertainty*, Proceedings of the 11th International Conference on Information Fusion, 2008.

[MR05] A. Mourikis and S. Roumeliotis, *Performance Bounds for Cooperative Simulataneous Localisation and Mapping (C-SLAM)*, Robotics: Science and Systems Conference, 2005.

[MT03] M. Montemerlo and S. Thrun, *Simultaneous localization and mapping with unknown data association using fastslam*, Proceedings of the IEEE International Conference on Robotics and Automation, 2003.

[MTKW02] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, *FastSLAM: A factored solution to the simultaneous localization and mapping problem*, Proceedings of the AAAI National Conference on Artificial Intelligence, 2002.

[MTKW03] _____, *Fastslam 2.0: An improved particle filtering algorithm for simultaneous localization and mapping that provably converges*, Proceedings of the International Joint Conference on Artificial Intelligence (ICJAI), 2003.

[MWBDW02] A. Makarenko, S. Williams, F. Bourgault, and H. Durrant-Whyte, *An experiment in integrated exploration*, Proceedings of the IEEE International Conference on Robotics and Systems, 2002.

[NCH06] P. Newman, D. Cole, and K. Ho, *Outdoor SLAM using Visual Appearance and Laser Ranging*, IEEE International Conference on Robotics and Automation, 2006.

[OG08] U. Orguner and F. Gustafsson, *Target Tracking Using Delayed Measurements with Implicit Constraints*, Proceedings of the 11th International Conference on Information Fusion, 2008.

[OG09] _____, *Distributed Target Tracking with Propagation Delayed Measurements*, Proceedings of the 12th International Conference on Information Fusion, 2009.

[RNDW07] F. Ramos, J. Nieto, and H. Durrant-Whyte, *Recognising and Modelling Landmarks to Close Loops in Outdoor SLAM*, IEEE International Conference on Robotics and Automation, 2007.

[SK95] R. Simmons and S. Koenig, *Probabilistic robot navigation in partically observable environments*, Proceedings of the International Joint Conference on Artificial Intelligence, 1995, pp. 1080–1087.

[SL95] R.L. Streit and T.E. Luginbuhl, *Probabilistic multi-hypothesis tracking*, IEEE Transactions on Aerospace and Electronic Systems (1995).

[SR05] R. Sim and N. Roy, *Global A-Optimal Robot Exploration in SLAM*, IEEE International Conference on Robotics and Automation, 2005.

[SSC90] R. Smith, M. Self, and P. Cheeseman, *Estimating uncertain spatial relationships in robotics*, Autonomous Robot Vehicles, 1990.

[Str00] R. L. Streit, *Tracking on intensity-modulated data streams*, Technical report 11221, NUWC, RI, USA, May 2000.

[Thr02] S. Thrun, *Exploring artificial intelligence in the new millenium, chapter 1 robot mapping: A survey*, 2002.

[TK99] S. Theodoridis and K. Koutroumbas, *Pattern recognition*, Academic Press, San Diego, 1999.

[TL05] S. Thrun and Y. Lui, *Multi-Robot SLAM with Sparse Extended Information Filters*, Robotics: Science and Systems Conference, 2005.

[TLK+01] S. Thrun, Y. Liu, D. Koller, A. Ng, Z. Ghahramani, and H. Durrant-Whyte, *Simultaneous localization and mapping with sparse extended information filters*, IEEE Transaction of Robotic and Automation, 2001.

[TM02] G. Theocharous and S. Mahadevan, *Approximate planning with hierarchical partially observable markov decision processes for robot navigation*, Proceedings of the IEEE International Conference on Robotics and Automation, 2002.

[WCN+09] B. Williams, M. Cummins, J. Neira, P. Newman, I. Reid, and J. Tardos, *A comparison of loop closing techniques in monocular slam*, Proceedings of the IEEE **57** (2009), no. 12, 1188–1197.

[WZ07] X. Wang and H. Zhang, *A UPF-UKF framework for SLAM*, IEEE International Conference on Robotics and Automation, 2007.

[Yam97] B. Yamauchi, *A frontier based approach for autonomous exploration*, Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, 1997.