Tracking and Classification with Wireless Sensor Networks and the Transferable Belief Model

Thesis submitted to Cardiff University in candidature for the degree of Doctor of Philosophy.

Matthew Simon Roberts

Cardiff School of Computer Science and Informatics
Cardiff University
December 2010

ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

DECLARATION

This work has not previously been accepted in substance for any degree and is not being concurrently submitted in candidature for any degree.

Signed ..M. Roberts..... (candidate) Date 24/10/2011.....

STATEMENT 1

This thesis is being submitted in partial fulfillment of the requirements for the degree of PhD.

Signed ..M. Roberts..... (candidate) Date 24/10/2011.....

STATEMENT 2

This thesis is the result of my own investigation, except where otherwise stated. Other sources are acknowledged by giving explicit reference.

Signed ..M. Roberts..... (candidate) Date ...24/10/2011.....

STATEMENT 3

I hereby give consent for my thesis, if accepted, to be available for photocopying and for inter-library loan, and for the title and summary to be made available to outside organisations.

Signed ....M. Roberts..... (candidate) Date ....24/10/2011.....

Abstract

The use of small, cheap, networked devices to collaboratively perform a task presents an attractive opportunity for many scenarios. One such scenario is the tracking and classification of an object moving through a region of interest. A single sensor is capable of very little, but a group of sensors can potentially provide a flexible, self-organising system that can carry out tasks in harsh conditions for long periods of time.

This thesis presents a new framework for tracking and classification with a wireless sensor network. Existing algorithms have been integrated and extended within this framework to perform tracking and classification whilst managing energy usage in order to balance the quality of information with the cost of obtaining it. Novel improvements are presented to perform tracking and classification in more realistic scenarios where a target is moving in a non-linear fashion over a varying terrain. The framework presented in this thesis can be used not only in algorithm development, but also as a tool to aid sensor deployment planning.

All of the algorithms presented in this thesis have a common basis that results from the integration of a wireless sensor network management algorithm and a tracking and classification algorithm — both of which are considered state-of-the-art. Tracking is performed with a particle filter, and classification is performed with the Transferable Belief Model.

Simulations are used throughout this thesis in order to compare the performance of different algorithms. A large number of simulations are used in each experiment with various parameter combinations in order to provide a detailed analysis of each algorithm and scenario.

The work presented in this thesis could be of use to developers of wireless sensor network algorithms, and also to people who plan the deployment of nodes. This thesis focuses on military scenarios, but the research presented is not limited to this.

Acknowledgements

Firstly, I would like to thank my supervisor Prof. David Marshall for all his support and guidance throughout my PhD. I would also like to thank Advanced Research Computing @ Cardiff, Cardiff University for the use of Condor, in particular Dr James Osborne for his technical assistance with Condor. Thank you also to my fellow lab members and PhD students for their helpful discussions and support.

I would like to acknowledge financial support from EADS. I would like to thank Dr Brian Turton for his support and helpful advice, Dr Gavin Powell for his technical assistance and advice, Peter Talbot-Jones for helping to create the scenarios used in this thesis, and all my other colleagues at EADS for their encouragement and support.

I would also like to thank my family and friends for their support throughout my PhD.

Finally, I would like to thank Becky for being so understanding and supportive throughout this lengthy process — I couldn't have done this without you!

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

bba basic belief assignment

bbm basic belief mass

cTBM Continuous Transferable Belief Model

DAG Directed Acyclic Graph

DST Dempster-Shafer Theory

DSmT Dezert-Smarandache Theory

EKF Extended Kalman Filter

HTC High-Throughput Computing

IDSQ Information-Driven Sensor Querying

IPB Intelligence Preparation of the Battlefield

JTAC Joint Tracking and Classification

LKF Linear Kalman Filter

MCR Matlab Compiler Runtime

PCR Proportional Conflict Redistribution

pdf probability density function

PDF probability distribution function

SMSE Sum of the Mean Squared Errors

SNR Signal to Noise Ratio

SIR Sampling Importance Resampling

SIS Sequential Importance Sampling

TBM Transferable Belief Model

UAV Unmanned Aerial Vehicle

VBA vacuous belief assignment

WSN Wireless Sensor Network

Chapter 1

Introduction

Wireless Sensor Networks (WSNs) present an attractive opportunity for the completion of many tasks. One such task is the estimation of an object's kinematic state and identity — that is, tracking and classification. This thesis is dedicated to integrating and improving upon the state-of-the-art in order to perform tracking and classification with a WSN. The research presented in this thesis develops an integrated solution for tracking and classification with a WSN whilst managing the energy usage of the nodes.

A Wireless Sensor Network is a group of small devices, each having an on-board processor and communication device, WSN nodes typically also have on-board sensors. WSN nodes are also known as *motes* or *smart dust* (Khan et al., 2000; Warneke et al., 2001) — although the latter usually refers to very small nodes. Each node typically has very limited resources and sensing ability. It is the group of nodes working together that makes a WSN an attractive technology — the system is greater than the sum of its parts.

A typical WSN node has many constraints, including:

- Limited energy storage
- Limited resources
- Short communication range
- Limited sensing ability

A typical general purpose WSN node, such as the MICA2 mote (Crossbow Technology Inc., 2006a), is powered by AA batteries and does not have a means of harvesting energy or recharging the batteries. WSN nodes typically switch off some components, such as the radio transceiver, to reduce energy usage. A mote can further reduce power consumption by *sleeping*. A sleep state is a low power state where most components are switched off, and the remaining components are inactive. Sleeping is essential to extend

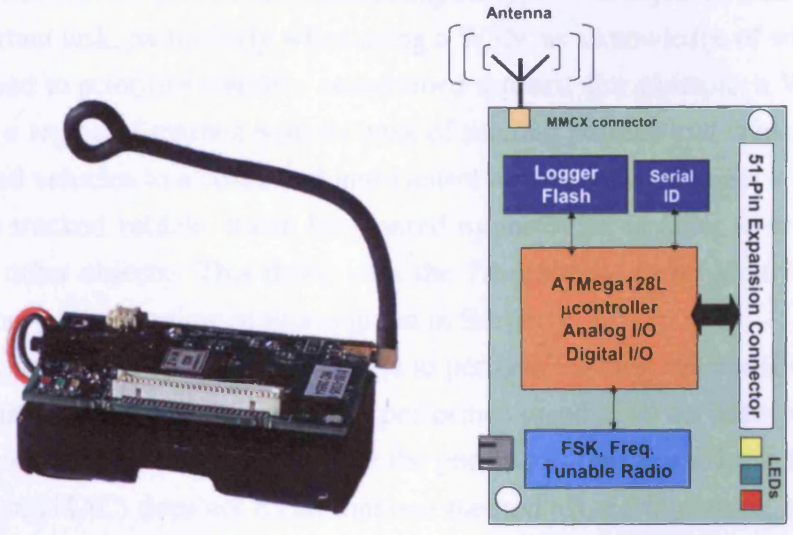
the life of a mote — the MICA2 mote (See Figure 1.1a) is stated as having a lifetime of more than one year on a single set of batteries by doing this (Crossbow Technology Inc., 2006a).

It should be noted that general purpose motes such as the MICA2 and TelosB (Crossbow Technology Inc., 2006c) are ideally suited to development and experimentation, but are not necessarily suited to real world applications. Applications that include harsh conditions, or necessitate lower costs might be better suited to bespoke hardware. However, general purpose motes have been used in harsh conditions by using a protective housing (PermaSense Project, 2010). Such motes would probably not be suited to use in military scenarios due to harsh electronic conditions, such as radio jamming, and tighter security constraints. It is not known whether bespoke motes have been created for real world military scenarios. The MICA2 and related motes have been used for examples due to their popularity and our experience with them.

The processor of a MICA2 mote, the ATmega128L (See Figure 1.1b), has a maximum speed of 8MHz (Atmel Corporation, 2010) and does not support floating point operations in hardware. It has only 128K bytes of program memory and 512K bytes of memory that can be used to log measurements (Crossbow Technology Inc., 2006a). The MICA2 mote has a maximum data rate of 38.4K baud. In order to use these limited resources in an efficient manner, operating systems such as TinyOS (TinyOS Community, 2010) have been developed. TinyOS is a light-weight operating system designed specifically for WSN devices with limited resources. A single executable runs on each node, which is created by integrating the operating system and the TinyOS program during compilation. In order to maximise the life of a WSN, nodes typically have synchronised periods of communication; if a node does not have a task to carry out then it will sleep between these periods.

The theoretical maximum range of the radio used by the 916MHz version of the MICA2 mote is approximately 150m (Crossbow Technology Inc., 2006a) — in practice it could be considerably less. We conducted empirical tests in an outdoor car park and found the maximum communication range to be an order of magnitude less than this. Due to the nature of radio communications, two nodes within a short range of each other are not guaranteed to have a reliable communications link. The maximum communication range is typically reduced due to less than ideal antenna configurations, attenuating obstacles such as walls and grass, and also by multipath propagation effects that occur from signals reflecting off objects and the ground.

A single WSN node is usually unable to provide sufficient sensing ability for an entire area of interest. The appeal of a WSN lies in the use of many cheap nodes to provide adequate coverage. For example, a WSN would be ideally suited to environmental



(a) (b)

Figure 1.1: A MICA2 mote (a) and its architecture (Crossbow Technology Inc., 2006b) (b)

monitoring of an office without the need for a wired communications system, or for tracking a vehicle by using a magnetometer in each node. For the latter example, each node can provide measurements about vehicle activity in the vicinity of the node. Such a system could be a more attractive option than using a small number of high quality sensors; for example, a camera-based system could be negatively affected by varying lighting conditions and poor weather.

Tracking, the estimation of an object's state from noisy observations, is an essential part of the algorithms developed within this thesis. Tracking algorithms can be used in a wide variety of applications such as in-car satellite navigation systems and missile guidance systems. The purpose of a tracking algorithm is to provide an estimate of a object of interest given a set of observations — typically this estimate is a probability distribution that indicates the uncertainty of the estimate. A brief overview of two popular tracking algorithms is given in Section 2.1.

The algorithms for tracking used in this thesis focus on tracking a single target at any one time. In a real world application, multi-target tracking might be considered essential, but it is not relevant to the core work of this thesis. To achieve multi-target tracking, not only would data association need to be considered, which could be difficult to achieve with range measurements alone, but it would also be necessary to modify the WSN management algorithm used throughout this thesis — this would not be a trivial task.

Classification is the process of determining the type of an object of interest. This can be an important task, particularly when using a WSN, as a knowledge of what an object is can be used to prioritise resource constrained sensors. For example, a WSN may be monitoring a region of interest with the task of sending position and velocity estimates of all tracked vehicles to a command and control centre; once an object is identified as not being a tracked vehicle, it can be ignored by the WSN in order to save resources or monitor other objects. This thesis uses the *Transferable Belief Model* (TBM) for classification — an overview of this is given in Section 2.2.2.

This thesis focuses on the use of WSNs to perform tracking and classification. Typically both tracking and classification are performed together — we refer to this as *Joint Tracking and Classification* (JTAC). For the purpose of this thesis, Joint Tracking and Classification (JTAC) does not mean that one method is used to perform both tracking and classification, it means that there is an inter-dependance between the two processes. For example, tracking may be performed using a particle filter, and the Transferable Belief Model (TBM) may be used for classification; however, a feedback loop could exist between the two processes — state estimates from the particle filter could be used for classification, and the classification output could be used to adjust the modelling within the particle filter.

Sections 1.1 and 1.2 will discuss the motivations and applications for tracking and classification with WSNs, respectively. An outline of the remainder of the thesis is given in Section 1.4. Section 1.3 states the main contributions of this thesis. A list of publications is given in Section 1.5.

1.1 Motivation

WSNs present many opportunities due to the properties they possess: mass production of nodes can result in low deployment costs; a carefully managed WSN can have a long lifetime; nodes can be deployed in different ways, including from an aircraft; and WSNs can be self-organising and can be re-programmed over the air.

A combination of these properties can result in an attractive option for the deployment of a WSN. For example, WSN nodes can be deployed for long periods of time in hazardous environments where it would be costly or hazardous to use humans. The loss of nodes would result in a relatively low cost and a self-organising network would adapt to the new topology. The nodes could be managed to sleep between events of interest — maximising the lifetime of the system. During deployment, the ability to remotely re-program a network of nodes allows an existing system to adapt to a new mission. Due to the low cost of the nodes, it may not be necessary to retrieve them once they

are no longer required. Existing and potential applications for WSN are discussed in Section 1.2.

WSNs present an interesting opportunity for tracking or JTAC scenarios. In such scenarios, a target can be tracked using nearby sensors. As the target moves through the region of interest, different sensors are used to monitor its movement — at any moment in time, the majority of node utilisation remains close to the target, allowing other nodes to sleep or perform another activity.

In addition to this, classification can be used to provide information about the types of objects in the region of interest. This information could be used to trigger alarms, and could be used to adapt the target tracking algorithm or parameters to better model the behaviour of the target. Chapters 4 and 5 both present similar but different methods for performing JTAC with a WSN.

1.2 Applications

WSNs have been used for various tasks including: sniper localisation (Maroti et al., 2004); dirty bomb detection and localisation (Vanderbilt University, 2008); wildlife monitoring (Szewczyk et al., 2004); smart homes (Hussain et al., 2009); flood detection (ALERT Systems, 2005). Potential applications vary greatly, including: health monitoring; the monitoring of hazardous areas or disaster zones with a lack of existing infrastructure; precision agriculture; and aiding navigation inside buildings (Akyildiz et al., 2002).

Within the context of tracking or tracking and classification, possible WSN applications include people tracking in buildings (Jamieson et al., 2004); monitoring movement in border zones; battlefield surveillance; vehicle direction and speed detection (UC Berkeley and MLB Co., 2001); and the tracking and classification of vehicles. For a more detailed review of WSN applications, see Akyildiz et al. (2002); García-hernández et al. (2007); Haenggi (2005); Shepherd and Kumar (2005). This thesis focuses on the tracking and classification of vehicles — particularly within a military context. Algorithms are not limited to this, but due sponsorship this has been the focus.

An interesting project conducted by UC Berkeley and MLB Co. (2001) was the estimation of a vehicle's kinematic state using motes deployed from a Unmanned Aerial Vehicle (UAV). Rene motes were used in the experiments along with 2 axis magnetometers. The Rene mote is the 3rd generation of Berkeley motes; the MICA mote, the predecessor to the MICA2 mote, is the 4th generation mote. The experiment successfully tackled many of the problems that face deployed WSNs, including time synchronisation

and multi-hop message routing, in order to detect military vehicles and estimate their velocity.

Instead of the above passive approach to tracking objects, the Mote Indoor Positioning System (Jamieson et al., 2004) locates people indoors using an active approach — each person wears a MICA2DOT that acts as a radio receiver for MICA2 nodes that act as beacons. The MICA2DOT is a smaller version of the MICA2 mote (Crossbow Technology Inc., 2007). The system uses the received signal strength indication of the radio communications to calculate relative distances to beacons.

Unfortunately such a system can be very inaccurate due to the unpredictable nature of radio frequency signals. A more accurate system for locating objects with the use of beacons can be achieved by using time of flight distance calculations — the Cricket mote implements this by using ultrasound (Crossbow Technology Inc., 2005). Alternatives to the cricket system that use standard MICA or MICA2 mote hardware include those by Sallai et al. (2004) and Whitehouse (2002).

1.3 Main Contributions

The main contributions of this thesis are:

- A novel framework for performing tracking or joint tracking and classification with a WSN — and its demonstrated use in both developing new algorithms and as a tool to aid sensor deployment.
- A novel method for jointly tracking and classifying a target with a WSN whilst managing the usage of sensors.
- Novel improvements to the above method to perform joint tracking and classification in more realistic scenarios by:
  - Considering, within the classification process, the uncertainty of the target's kinematic state, the terrain, and how terrain restricts a target's movement.
  - Adding an 'intelligent memory" to the TBM to improve the output of classifications that are updated iteratively over time; preventing the assignment of belief to target classes that past behaviour has shown are not feasible.

1.4 Thesis Outline

The thesis is organised as follows:

- Chapter 2 provides an overview of relevant topics. Topics discussed include tracking, classification and sensor management — all of which are essential to the research presented in the later chapters of this thesis.
- Chapter 3 presents the framework in which the algorithms have been implemented. Further chapters utilise this framework to compare the performance of algorithms.
- Chapter 4 presents a novel method to jointly track and classify a target with a WSN, whilst also planning the usage of sensors in a such a way that balances the cost of communications with the quality of information obtained by the sensors. This chapter is an expanded version of Roberts and Marshall (2008).
- Chapter 5 improves upon Chapter 4 for use with more realistic scenarios. Novel improvements include the use of terrain information, the use of an elliptical area of a terrain map to account for position uncertainty and its subsequent use within the classification stage, and the addition of an 'intelligent memory' within the classification. A condensed version of this chapter was presented at the International Conference on Information Fusion 2010 (Roberts et al., 2010).
- Chapter 6 shows how the framework presented in Chapter 3 can be used as a tool to aid sensor deployment planning. This is illustrated with a comparison of WSN Joint Tracking and Classification performance using two different sensor deployment strategies.
- Chapter 7 summarises the work presented in this thesis and discusses possible future work.

1.5 List of Publications

- Roberts, M., Marshall, D., and Powell, G. (2010). Improving joint tracking and classification with the Transferable Belief Model and terrain information. In *Proceedings of International Conference on Information Fusion 2010*, Edinburgh, UK. Jul 2010.
- Powell, G., Roberts, M., and Marshall, D. (2010a). Empty set biasing issues in the Transferable Belief Model for fusing and decision making. In *Proceedings of International Conference on Information Fusion 2010*, Edinburgh, UK. Jul 2010.

- Powell, G., Roberts, M., and Owen, T. (2010c). Transferable belief models for human welfare. In *Proceedings of International Conference on Information Fusion 2010*, Edinburgh, UK. Jul 2010.

- Powell, G., Roberts, M., and Marshall, D. (2010b). Pitfalls for recursive iteration in set based fusion. In *Workshop on the Theory of Belief Functions*, Brest, France. Apr 2010.

- Roberts, M. and Marshall, D. (2008). Using classification to improve wireless sensor network management with the continuous transferable belief model. In Carapezza, E. M., editor, *Unmanned/Unattended Sensors and Sensor Networks V*, volume 7112, page 711204, Cardiff, UK. SPIE. Sep 2008.

Chapter 2

Background

This chapter gives a brief overview of various topics that relate to the research presented in this thesis. The topics can be broadly split into four sections: tracking, classification, information theory, and sensor management. Tracking is the process of estimating the kinematic state of an object of interest using noisy observations. For classification, we focus on the Transferable Belief Model — a way of modelling subjective beliefs. Information theory is discussed as it is a vital part of sensor management; sensors are selected on the basis of the reduction in uncertainty that an observation will provide. Sensor management is a key part of any WSN application. If a WSN is not adequately managed then it is likely to either under perform, for example in tracking accuracy, or place too great a demand on limited resources.

2.1 Tracking

Tracking involves the estimation of an object's kinematic state, this could be its current state or its future state. Noisy measurements of an object's state over a period of time are used in the estimation process. Filtering is the process of using the measurements to estimate the current object state, as opposed to estimating the future object state (prediction) (Kalman, 1960).

The ability to track an object and predict its future location can be an invaluable feature of a WSN. For example, when an object of interest is moving through a WSN, the estimate of the future path of the object can be used to decide which sensors should be active to continue the observation of the object. For example, Williams et al. (2007) use tracking in the process of selecting a leader node for a WSN.

Two popular types of methods for tracking are discussed here, *Kalman filtering* and *particle filtering*. The Kalman filter is the optimal method for tracking an object that has linear dynamics, and Gaussian process and measurement noise. A particle filter can

be used to track an object with non-linear dynamics, a non-Gaussian state estimate, and non-Gaussian noise. Both the Kalman filter and particle filter are Bayesian estimators. They are also recursive — the object's state $x_k \in \mathbb{R}^n$ at time k, is based upon the objects previous states; this enables an iterative solution to be used which updates the current estimate using all of the past measurements (Welch and Bishop, 2004).

The discrete time versions of the Kalman and particle filters will be discussed in the remainder of this section. For each time step, both Kalman and particle filters have a *predict* and an *update* step (Doucet et al., 2001a; Welch and Bishop, 2004). The predict step (Also known as the time update step) is used to project the estimate to the current time step, producing the *a priori* estimate. The update step (Also known as the measurement update step) is used to condition the a priori estimate using the observation from the current time step, resulting in the *a posteriori* estimate.

2.1.1 Kalman Filter

The Kalman filter (Kalman, 1960) models the object's state probability density function (pdf) as a Gaussian — parameterised by a state estimate (the mean of the Gaussian) and a posteriori estimate error covariance (the variance of the Gaussian) (Maybeck, 1979, chap. 1). Both the system dynamics and the observation model must be linear, and both the process and observation noise must be Gaussian — the Kalman filter is the optimal filter for such cases. The Kalman filter usually requires considerably less memory and computational power than particle filtering.

The target state evolves using the following linear model (Welch and Bishop, 2004):

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} + \mathbf{w}_{k-1}, \tag{2.1}$$

where $\mathbf{F}$ is a $n \times n$ matrix that updates the target's state at time $k-1$ to time k, $\mathbf{u}$ is an optional control input, $\mathbf{B}$ relates the control input to the target state, and $\mathbf{w}_{k-1}$ is the process noise. The observation for time step k, $\mathbf{z}_k \in \mathbb{R}^m$, is calculated using

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{v}_k, \tag{2.2}$$

where $\mathbf{H}$ is an $m \times n$ matrix that relates the target state to the observation, and $\mathbf{v}_k \in \mathbb{R}^m$ is the measurement noise.

Variations of the Kalman Filter exist, including the Linear Kalman Filter (LKF) and the Extended Kalman Filter (EKF); both of which linearise a non-linear system about a nominal point using a Taylor series expansion (Williams, 2007). The LKF uses a fixed nominal point, whereas the EKF uses the current estimate as the nominal point. A

problem with both the LKF and EKF is that the probability distributions are no longer Gaussian once their respective random variables have been linearised. In addition to this, neither of the two aforementioned filters may be suitable for tracking a target with a WSN in certain scenarios. For example, if range sensors are used the probability distribution of the target's state can become significantly non-Gaussian — particularly as a target approaches a sensor.

2.1.2 Particle Filter

The particle filter (Arulampalam et al., 2002; Doucet et al., 2001a; Gordon et al., 1993) approximates an object's state pdf using samples (particles); a consequence of this is that the probability distribution function (PDF) is not limited to a Gaussian distribution. It is possible to use both a non-linear target dynamics and a non-linear observation model. Two well-known variations of the Particle Filter are the *Sampling Importance Resampling (SIR) filter* and the *Sequential Importance Sampling (SIS) filter*.

In contrast to the Kalman filter, the target state evolves using a model which can be non-linear (Arulampalam et al., 2002):

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}, \mathbf{w}_{k-1}), \tag{2.3}$$

where $\mathbf{f}(.)$ is a function that updates the target from time step $k-1$ to step k. The observation model can also be non-linear:

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k, \mathbf{v}_k), \tag{2.4}$$

where $\mathbf{h}(.)$ relates the target state to the observation. It is possible to vary both the state update and observation functions at each time step, but in this thesis we use the same functions for every time step.

The a posteriori density of the estimated state at time k, $\mathbf{x}_k$, is approximated by a set of N_p weighted particles (Ristic et al., 2004a, p. 38):

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N_p} w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i), \tag{2.5}$$

where w_k^i is the *importance weight* (or 'weight') of the i^{th} particle $\mathbf{x}_k^i$, and $\delta(.)$ is the Dirac delta function.

2.1.2.1 Sampling Importance Resampling

The Sampling Importance Resampling (SIR) filter (Gordon et al., 1993) is possibly the simplest Particle Filter. At the predict step, each particle is drawn from the *proposal distribution*:

$$\mathbf{x}_k^i \sim p\left(\mathbf{x}_k|\mathbf{x}_{k-1}^i\right) \tag{2.6}$$

with uniform weight, $w_k^i = 1/N_p$. This forms the a priori estimate of the target state at time k. The symbol $\sim$ denotes sampling from a distribution.

At the update step, each particle of the a priori estimate is compared with the observation, and it's weight is calculated (Doucet et al., 2001a):

$$w_k^i \propto p\left(\mathbf{z}|\mathbf{x}_k^i\right), \tag{2.7}$$

where $\mathbf{x}_k^i$ is particle $i = 1,...,N_p$ at time k, $\mathbf{z}_k$ is the observation at time k, and $a \propto b$ denotes that a is proportional to b. The particle weights are normalised such that $\sum_{i=1}^{N} w_k^i = 1$.

A new set of N particles is then sampled, with the probability of a particle being selected equal to it's importance weight. Particles with low weights have a higher probability of being discarded, and particles with high weights have a higher probability of being duplicated. The new particles are the a posteriori estimate for time step k. The elimination of particles with low weights helps to prevent *degeneracy*; this occurs when a small number of particles have a large weights and the remaining particle have very small weights.

2.1.2.2 Sequential Importance Sampling

Sequential Importance Sampling (SIS) can be seen as a more generalised version of the SIR filter. A different proposal distribution is used, importance weights are calculated differently, resampling does not have to take place at every time step, and the probability that a particle will be selected during resampling is not restricted to it's weight.

The proposal distribution $p\left(\mathbf{x}_{k+1}|\mathbf{x}_k^i, \mathbf{z}_{k+1}\right)$ is used, it is common to use the notation $q(.)$ (e.g. Ristic et al., 2004b) or $\pi(.)$ (e.g. Doucet et al., 2001a) for this distribution — we will use the former throughout the remainder of this thesis. In this thesis we approximate the proposal distribution using the EKF update equations as is suggested by Williams et al. (2007) — this is discussed in more detail in Chapter 4. Importance weights are calculated using (Doucet et al., 2001a, pp. 9–10):

$$w_k^i \propto w_{k-1}^i \frac{p\left(\mathbf{z}_k|\mathbf{x}_k^i\right) p\left(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i\right)}{q\left(\mathbf{x}_k^i|\mathbf{x}_{k-1}^i, \mathbf{z}_k\right)}. \tag{2.8}$$

If resampling took place at time step $k-1$ then $w_{k-1}^i = 1/N_p$, and due to normalisation it will have no resultant effect on w_k^i. If resampling did not take place at time step $k-1$, then w_{k-1}^i will alter w_k^i.

The SIS filter does not require resampling to take place at every time step. Typically, resampling will take place when the number of effective particles, a measure of degeneracy, drops below an arbitrary threshold. An estimate for the number of effective particles, $\bar{N}_{eff}$, is calculated using (Kong et al., 1994; Ristic et al., 2004a, pp. 40-41):

$$\bar{N}_{eff} = \frac{1}{\sum_{i=1}^{N} \left(w_t^i\right)^2} \tag{2.9}$$

The probability that a particle is selected during resampling is $a^{(j)}$. For SIR, $a^{(j)} = w_k^i$. With SIS, $a^{(j)}$ is not limited to the particle weight. As it is desired to duplicate particles with high weights and to discard particles with low weights, $a^{(j)}$ should be a monotonically increasing function of w_k^i (Liu et al., 2001, p. 230). In this thesis, we use $a^{(j)} = w_k^i$ with SIS.

Alternatives to resampling, such as *reallocation*, can be used but are not necessary for the research conducted in this thesis as resampling is sufficient. Reallocation duplicates particles where $a^{(j)} \geq 1$, dividing the old particle's weight equally between the new copies. Particles with $a^{(j)} < 1$ are removed with a probability $1 - a^{(j)}$, the surviving particle's weights are increased in proportion to $a^{(j)}$. Reallocation tends to change the number of particles (Liu et al., 2001, p. 231) — although a strategy exists to fix the number of particles.

2.1.2.3 Systematic Resampling

Most of the particle filter configurations for this thesis use systematic resampling ([19] in Ristic et al. (2004a, chap. 3)). Systematic resampling is simple to implement and its computational complexity is $O(N_p)$. The systematic resampling algorithm is shown in Algorithm 2.1, $u \sim \mathcal{U}[a,b]$ denotes drawing from a uniform random pdf within the interval $[a,b]$. Alternatives to systematic resampling include residual resampling and stratified resampling; for more details on resampling see Douc et al. (2005).

The aim of systematic resampling, as with other resampling schemes, is to reduce degeneracy. A potential side-affect of this resampling scheme is that particles with high weights can be sampled many times leading to a lack of diversity — this is termed *sample impoverishment* (Arulampalam et al., 2002).

Algorithm 2.1 Systematic resampling

$\left[\left\{\mathbf{x}_k^{j*}, w_k^{j*}\right\}_{j=1}^{N_p}\right] = \textbf{SystematicResample}\left[\left\{\mathbf{x}_k^i, w_k^i\right\}_{i=1}^{N_p}\right]$

c = cumulativeSum$\left(\left\{w_k^i\right\}_{i=1}^{N_p}\right)$ {Construct cumulative sum of weights}
$i = 1$ {Cumulative sum index}
$u \sim \mathcal{U}\left[0, \frac{1}{N_p}\right]$ {Point along cumulative sum}
for $j = 1,..,N_p$ **do**
  while $u < c_i$ **do**
    $i = i + 1$
  end while
  $\mathbf{x}_k^{j*} = \mathbf{x}_k^i$ {Assign sample}
  $w_k^{i*} = \frac{1}{N_p}$ {Assign weight}
  $u = u + \frac{1}{N_p}$ {Move along cumulative sum}
end for

2.1.3 Data Fusion for Tracking

Data fusion is the aggregation of sensor data from multiple sensors, or modalities, or both. Using data from more than one 'view' can increase the Signal to Noise Ratio (SNR) for tracking. Data fusion is very useful in WSNs because measurements, possibly of a low resolution, can be taken from many nodes, the combination of these measurements can provide a more accurate state estimate.

Various WSN applications utilise data fusion for target tracking (Arora et al., 2004; Brooks et al., 2003; Williams et al., 2007). It should be noted that data fusion differs from *decision fusion* (Brooks et al., 2003). Data Fusion is the fusion of sensor data, decision fusion is the fusion of decisions, estimations, feature vectors, or models. To achieve data fusion over multiple nodes within a WSN, sensor data must be sent from multiple nodes to a single node (the leader node) where the sensor data is fused. To achieve decision fusion over multiple nodes within a WSN, sensor data is used in local computation at each node, the results of the computation are then sent to the leader node where it is fused. Decision fusion can require less data to be sent to the leader node, and for less computation to take place at the leader node, but in some circumstances, data fusion can produce better results (Brooks et al., 2003). The work presented in this thesis builds upon work by Williams et al. (2007). Consequently, we perform data fusion at a leader node (which changes over time) — decision fusion is therefore effectively out of the scope of the work presented within this thesis.

Williams et al. (2007) achieves data fusion by combining the measurements in the EKF equations within the SIS filter. A measurement for the SIS consists of a vector

of range measurements, one for each active sensor. This is discussed in more detail in Section 4.1.2.

Another possibility for the fusion of observations from multiple sensors within a particle filter would be to use different particles for each sensor measurement. At each step a portion of the particles would be updated using the observation from sensor s — resulting in a vote; observations that are similar would agree, producing a higher peak in the PDF than peaks produced by observations from sensors which are very different to observations from other sensors.

Qi et al. (2001) discuss the usage of mobile agents to perform data fusion in a WSN. Mobile agents could be used to perform data fusion for target tracking. However, it is not clear if target tracking could be performed in real-time using mobile agents — it may take longer for an agent to traverse nodes in the WSN and then perform data fusion compared to the length of time it would take for a leader to receive sensor data and perform the data fusion.

2.2 Statistical Classification

Classification is the process of distinguishing what an object is. It is important to classify objects that are detected in a WSN; classification can be used to prioritise targets, trigger alarms, or to aid object association in multi-target sensor networks.

The remainder of this section will briefly discuss Bayesian probability theory and then discuss the TBM. Two approaches that are related to the TBM, Dempster-Shafer Theory (DST) and Dezert-Smarandache Theory (DSmT), will also be briefly discussed. The aim of this section is not to present the TBM as a better approach to Bayesian probability; it is simply to discuss an alternative to the Bayesian approach that is ideally suited to the research undertaken in this thesis. Many other methods for classification exist — these are not discussed as they are not relevant to the approach taken in this thesis.

2.2.1 Bayesian Probability Theory

Bayesian probability theory (Bayes, 1763), named after Rev. Thomas Bayes, provides a rigourous mathematical approach for modelling and calculating probabilities. Probabilities are assigned to mutually exclusive hypotheses. Bayes' theorem provides a means of updating the probability of an event given a test or an observation — this is a key part of Bayesian probability theory.

A probability is assigned to each mutually exclusive outcome $\omega \in \Omega$:

$$0 \leq P(\omega) \leq 1 \; \forall \omega \in \Omega, \tag{2.10}$$

and

$$\sum_{\omega \in \Omega} P(\omega) = 1. \tag{2.11}$$

Bayes' theorem for discrete probabilities is:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \tag{2.12}$$

where $P(A)$ is the *prior* probability of event A, $P(B|A)$ is the *likelihood*, $P(B)$ is the *marginal* probability of event B, and $P(A|B)$ is the *posterior* probability. Bayes' theorem can be seen as a way of updating the (prior) probability of an event A given a test for the event with an accuracy of $P(B|A)$. $P(B)$ would be calculated as $P(B|A)P(A) + P(B|\bar{A})P(\bar{A})$ — this is the probability of B given all hypotheses; the marginal normalises the posterior probability.

The posterior probability of event A given n observations can be calculated using (Robert, 2007, chap. 1):

$$P(A|y_1, y_2, \ldots, y_n) = \frac{P(y_1, y_2, \ldots, y_n|A)P(A)}{P(y_1, y_2, \ldots, y_n)}. \tag{2.13}$$

For example, suppose we have a target class, Bicycle, we can calculate the probability of an object belonging to the class Bicycle given n observations of its speed, $s_1, s_2, \ldots, s_n$, using:

$$P(Bicycle|s_1, s_2, \ldots, s_n) = \frac{P(s_1, s_2, \ldots, s_n|Bicycle)P(Bicycle)}{P(s_1, s_2, \ldots, s_n)} \tag{2.14}$$

In can be difficult to calculate the likelihood and marginal joint probability distributions — especially for a large number of outcomes. If the observations are independent, then since

$$P(y_1, y_2, \ldots, y_n) = P(y_1)P(y_2)\ldots P(y_n) \text{ and}$$
$$P(y_1, y_2, \ldots, y_n|A) = P(y_1|A)P(y_2|A)\ldots P(y_n|A),$$

the above equation can be simplified to

$$P(A|y_1, y_2, \ldots, y_n) = \frac{P(A)\prod_{i=1}^{n} P(y_i|A)}{\prod_{i=1}^{n} P(y_i)}. \tag{2.15}$$

Unfortunately, making such an assumption can be a vast over simplification of the problem. Bayesian probability theory requires that the problem be adequately modelled — this can be a difficult task for complex problems. For example, continuing with the above example, we could assume that observations are independent, simplifying Equation 2.14 to:

$$P(Bicycle|y_1, y_2, \ldots, y_n) = \frac{P(Bicycle)\prod_{i=1}^{n} P(s_i|Bicycle)}{\prod_{i=1}^{n} P(s_i)}. \tag{2.16}$$

However, the above equation is not necessarily suitable for observations that are not independent. For example, it might not be suitable for scenarios where a system gives successive observations over time.

2.2.2 The Transferable Belief Model

The *Transferable Belief Model* (TBM) (Smets and Kennes, 1994) is used for modelling subjective beliefs; it is capable of handling uncertainty, ignorance, conflicting information, and an open-world — where belief is given to outcomes outside of the set of hypotheses. It is an extension of DST (Dempster, 1968; Shafer, 1976) which is a generalisation of Bayesian Theory. Two levels are used in the TBM — the *credal* and *pignistic* levels. The credal level is where beliefs are quantified, updated, and combined. At the the pignistic level, probability functions are created from the credal level — these probability functions can be used for classification and decision making. With the Continuous Transferable Belief Model (cTBM), it is possible to assign belief in continuous spaces, and to assign beliefs induced by a continuous pdf (Ristic and Smets, 2004; Smets, 2005). Subjective beliefs from different agents can be fused using various combination rules. TBM does not require an underlying probability model and prior probabilities are not required — providing a convenient alternative to Bayesian probability theory where both are required.

The TBM provides an alternative to Bayesian Theory; it easily allows the use of open worlds by assigning belief to the empty set. It also provides the ability to assign belief to a set of hypotheses rather than just a singleton hypothesis. These two features allow for the modelling of uncertainty in an intuitive manner. It should be noted that it is also possible to model uncertainty and open worlds within a Bayesian approach — Maskell (2008) discusses this, along with a comparison with the approach of the TBM.

The remainder of this section will discuss the TBM in more detail. The fundamentals of the TBM will be introduced, as well as the cTBM. A number of fusion rules are discussed. DST and DSmT will briefly be contrasted and compared with the TBM.

2.2.2.1 Credal Level

In the credal level, the quantified beliefs of an agent are modelled on a given frame of discernment $\Omega = \{\omega_1, \omega_2, \ldots, \omega_n\}$; each ω is a possible outcome, one of which is the the true state, $\bar{\omega}$. The empty set, $\emptyset$, is used to show conflict between beliefs and can be used for open worlds; open worlds allow us to consider outcomes outside of our frame of discernment. An agent quantifies how much support it gives to a subset of possible outcomes with a *basic belief mass (bbm)*,

$$m : 2^{\Omega} \rightarrow [0,1] \text{ with } \sum_{A \subseteq \Omega} m(A) = 1. \tag{2.17}$$

For example, if $\Omega = \{Bicycle, Car, Tank\}$, then belief can be assigned to the elements of

$$\begin{aligned} 2^{\Omega} = \{ &\{Bicycle\}, \{Car\}, \{Tank\}, \\ &\{Bicycle, Car\}, \{Car, Tank\}, \{Bicycle, Tank\}, \\ &\{Bicycle, Car, Tank\}, \emptyset\}. \end{aligned} \tag{2.18}$$

In a closed world, if you think that your object of interest is probably not a Tank, but you are not sure if it is a Bicycle or a Car then you might assign the following *bbms*:

$$\begin{aligned} m(\{Tank\}) &= 0.1 \\ m(\{Bicycle, Car\}) &= 0.9. \end{aligned} \tag{2.19}$$

In closed-world scenarios, $m(\emptyset) = 0$. In contrast, in open-world scenarios, $m(\emptyset) \geq 0$.

The *degree of belief* of A, $bel(A)$, quantifies our belief that $\bar{\omega} \subseteq A$, and is calculated as follows:

$$bel : 2^{\Omega} \rightarrow [0,1] \text{ with } bel(A) = \sum_{\emptyset \neq B \subseteq A} m(B) \qquad \forall A \subseteq \Omega, A \neq \emptyset. \tag{2.20}$$

The *degree of plausibility* of A, $pl(A)$, is calculated as follows:

$$pl : 2^{\Omega} \rightarrow [0,1] \text{ with } pl(A) = \sum_{B \cap A \neq \emptyset} m(B) = bel(\Omega) - bel(\bar{A}), \tag{2.21}$$

where $\bar{A}$ is the complement of set A. The *bbms* included in $pl(A)$ are masses that give potential support to A. The degree of plausibility is another way of presenting the same information as that provided by the degree of belief (Smets and Kennes, 1994).

The *bbms* for all of the outcomes in 2^{Ω} are collectively known as a *basic belief assignment (bba)*, i.e. a *bba* is a set of *bbms*. A *focal set*, A, is a set which has some support, $m(A) > 0$. Total ignorance can be represented by $m(\Omega) = 1$ — this is known as a *vacuous belief assignment (VBA)* (also known as a *vacuous belief function*).

2.2.2.2 Pignistic Level

In the pignistic level, the *pignistic transform* is used to calculate the pignistic probability function, *BetP*. For closed worlds, this is calculated as follows (Smets and Kennes, 1994):

$$BetP(B) = \sum_{A \subseteq \Omega} m(A) \frac{|A \cap B|}{|A|} \qquad \forall A \subseteq \Omega, \tag{2.22}$$

where $|A|$ is the size of set A. For open worlds, the following can be used (Smets, 2000):

$$BetP(B) = \sum_{A \subseteq \Omega} \frac{m(A)}{1 - m(\emptyset)} \frac{|A \cap B|}{|A|}. \tag{2.23}$$

It can be seen that if $m(\emptyset) = 0$ then Equations 2.22 and 2.23 are equivalent.

For example, if we assume a closed world and continue using the frame of discernment from the example in Section 2.2.2.1, then $BetP(\{Car\})$ is calculated as follows:

$$\begin{aligned} BetP(\{Car\}) &= \sum_{A \subseteq \Omega} m(A) \frac{1}{|A|} \\ &= m(\{Car\}) + \frac{m(\{Bicycle, Car\})}{2} + \frac{m(\{Car, Tank\})}{2} + \\ &\quad \frac{m(\{Bicycle, Car, Tank\})}{3}. \end{aligned} \tag{2.24}$$

The reasoning behind the pignistic transform is that of the *insufficient reason principle*; if a probability distribution must be built on n elements, given a lack of information, the probability should be equal for each element. The pignistic transform does this for each basic belief mass (Smets and Kennes, 1994).

2.2.2.3 The Continuous Transferable Belief Model

Using the *cTBM* it is possible to assign belief to masses conditional on prior probabilities (Caron et al., 2006; Delmotte and Smets, 2004; Ristic and Smets, 2004; Smets, 2005):

$$m(A|x) = \prod_{c_i \in A} pl(x|c_i) \prod_{c_i \in \bar{A}} [1 - pl(x|c_i)], \tag{2.25}$$

where $m(A|x)$ is the mass assigned to A given x, c_i is an element of A, $\bar{A} = 2^{\Omega} \backslash A$. The plausibility of x for a given class (e.g. c_i) can be calculated for a 'bell shaped'[1] *pignistic density function*, $Betf$ using (Ristic and Smets, 2004):

$$pl(x) = (x - \bar{x})\, Betf(x) + \int_x^{\infty} \left(1 - \frac{\mathrm{d}\bar{a}}{\mathrm{d}a}\right) Betf(a)\ \mathrm{d}a, \tag{2.26}$$

where $\bar{x}$ is defined as $Betf(x) = Betf(\bar{x})$ and $\bar{x} \leq v \leq x$; v is the mode of $Betf$. It is also posssible to calculate masses conditional on multi-modal densities (Doré and Martin, 2010) but this is not required for the work presented in this thesis and is therefore not discussed in further detail.

Ristic and Smets (2004) state the that the pignistic transform can then be calculated as follows:

$$BetP\{c_i|x\} = \sum_{A:c_i \in A} \frac{1}{|A|} \frac{m(A|x)}{[1 - m(\emptyset|x)]}. \tag{2.27}$$

As c_i is a singleton, it can be seen that the above equation is equivalent to Equation 2.23. This has been used for target classification of aircraft conditional on the observed speed of the target (Powell et al., 2006). It is also used in this thesis for classification of ground targets.

2.2.2.4 Fusion of basic belief assignments

This section aims to give a brief overview of fusion rules relevant to this thesis; a more comprehensive overview of fusion and conflict resolution can be found in Martin and Osswald (2007); Osswald and Martin (2006); Smarandache and Dezert (2005); Smets (2007). Various methods for fusion exist, there are usually based on *conjunctive combination* (also known as conjunctive consensus), *disjunctive combination*, or a mixture of the two. Conjunctive combination, denoted as $m_{\cap}$, is calculated as follows:

$$m_{1 \cap 2}(X) = \sum_{\substack{A,B \in 2^{\Omega} \\ A \cap B = X}} m_1(A)\, m_2(B), \tag{2.28}$$

[1] According to Smets (2005):

> A 'bell shaped' density is a unimodal density, continuous and strictly monotone increasing (decreasing) at left (right) of the mode

where m_1 and m_2 are *bbms* from agents 1 and 2 respectively. The total conflict between agents 1 and 2, k_{12}, is calculated using:

$$k_{12} = \sum_{\substack{A,B \in 2^\Omega \\ A \cap B = \emptyset}} m_1(A) m_2(B). \tag{2.29}$$

The conjunctive rule of combination can be seen as a way of fusing *bbas* that are reliable. When combinations take place using this rule, belief can be seen being 'pushed' towards singleton sets and k_{12}. When the *bba* is normalised (See Dempster's rule of combination on 22), the conjunctive rule can be seen as a combination rule that results in a more certain belief assignment.

Disjunctive combination is calculated as follows (Dubois and Prade, 1988):

$$m_{1 \cup 2}(X) = \sum_{\substack{A,B \in 2^\Omega \\ A \cup B = X}} m_1(A) m_2(B). \tag{2.30}$$

In contrast to the conjunctive rule, this rule can be seen as a way of fusing *bbas* where one is considered reliable and the other is not — but it is not known which agent is the reliable one. Conflict is 'pushed' towards the higher cardinality sets — and ultimately the ignorant set.

It is usually desirable for a combination rule to resolve or redistribute conflict in sensible manner. For example, if $\Omega = \{A, B, C, D\}$, and we have the following focal sets for two agents, 1 and 2:

$$m_1(A) = 0.99$$
$$m_1(B) = 0.01$$
$$m_2(B) = 0.01$$
$$m_2(C) = 0.99$$

If we combine the beliefs of agents 1 and 2 using some combination rule then it is intuitive that for the combined *bba* we should have $m(D) = 0$ and the following should probably *not* be the case:

$$m(A) = 0$$
$$m(B) = 1$$
$$m(C) = 0$$

It is important to note that the above result would occur if a closed-world combination

rule was used that assumed both agents were reliable — the above *bbas* indicate that a closed-world assumption cannot be used, one of the agents is not reliable, or that both of these constraints have been broken.

Fusing the *bbas* of agents can be a computationally intensive task — particularly when a frame of discernment has a high cardinality. One possible approach to reducing this burden is to constrain the number of focal sets in a belief assignment. One method to achieve this is to approximate a *bba* using only singleton focal sets (Voorbraak, 1990) — this results in approximation that is the same as a Bayesian probability assignment.

Three other methods approximate the original *bba* by keeping the a number of the highest-valued focal sets, and redistributing the remaining mass. *k-l-x* redistributes the remaining mass by normalising the new *bba* ([12] in Bauer (1997)). The Summarization method (Lowrance et al., 1986) redistributes the remaining mass to the set theoretic union of the elements that are discarded. D1 improves upon this by redistributing mass in a more sophisticated manner (Bauer, 1997).

Dempster's Rule

Dempster's rule of combination, denoted m_{DS}, is a normalised version of conjunctive combination. Dempster's rule of combination for combining two agents' *bbas* is calculated as follows (Shafer, 1976):

$$m_{DS}(X) = \frac{1}{k_{12}} \cdot \sum_{\substack{A,B \in 2^{\Omega} \\ A \cap B = X}} m_1(A) m_2(B), \tag{2.31}$$

$m_{DS}(\emptyset) = 0$, and k_{12} is calculated using Equation 2.29. This rule of combination can only be applied with closed worlds. Conflict is redistributed via normalisation — it does not take into account what caused the conflict.

Smets' Rule

Smets' rule of combination, denoted m_S, is calculated using conjunctive combination (Smets, 1990):

$$m_S(X) = \begin{cases} m_{1 \cap 2}(X) & \text{if } X \in 2^{\Omega}; \\ k_{12} & \text{if } X = \emptyset. \end{cases} \tag{2.32}$$

This rule does not proportionally redistribute conflict, it simply assigns it to the empty set — in a closed world this represents conflict between agents 1 and 2; in an open world, $m_S(\emptyset) > 0$ could signify belief for an outcome that is not in 2^{Ω}.

PCR

A more sophisticated method for fusing *bbas* called PCR5, has been presented by Smarandache and Dezert (2005). PCR5, part of a family of Proportional Conflict Redistribution (PCR) rules, is a more correct way of redistributing conflict than Dempster's rule as it only redistributes mass to outcomes involved in conflict. The PCR5 rule for combining beliefs from two agents, 1 and 2, is calculated using:

$$m_{PCR5}(X) = m_{1\cap 2}(X) + \sum_{\substack{Y\in 2^{\Omega}\setminus\{X\} \\ X\cap Y=\emptyset}} \left[\frac{m_1(X)^2 m_2(Y)}{m_1(X)+m_2(Y)} + \frac{m_2(X)^2 m_1(Y)}{m_2(X)+m_1(Y)} \right], \tag{2.33}$$

where $m_{1\cap 2}$ is Equation 2.28. All fractions with a denominator equal to zero are discarded. An alternative rule to PCR5 is PCR6 (Martin and Osswald, 2006). PCR6 is identical to PCR5 for 2 agents, but provides more intuitive results for 3 or more agents (Smarandache and Dezert, 2006b). In this thesis, we only fuse beliefs from 2 agents at any one time; since the result is the same using either PCR5 or PCR6, we will use the term PCR to refer to this.

Discounting

Discounting can be used prior to combining *bbas* where the reliability of a belief assignment is known. Discounted masses, m^{α}, are calculated as follows (Smets, 1993):

$$m^{\alpha}(A|x) = \begin{cases} \alpha m(A|x) & \text{if } A \subset \Omega; \\ \alpha m(A|x) + (1-\alpha) & \text{if } A = \Omega, \end{cases} \tag{2.34}$$

where a higher value of α indicates a more reliable source. α must be within the interval $[0,1]$ — where $\alpha = 0$ means completely unreliable, and $\alpha = 1$ represents 100% reliable.

When $\alpha < 1$, the above equation creates a more vague belief assignment — reducing conflict. An extension to discounting, which takes into more detailed information of an agents reliability, can be found in Mercier et al. (2005).

2.2.2.5 Belief Functions

The TBM is part of a family of approaches – termed *belief functions*. Two alternative popular belief function approaches include Dempster-Shafer Theory (DST) and Dezert-Smarandache Theory (DSmT). All of the approaches have many similarities since the TBM and DSmT are extensions of DST. This section will briefly discuss DST and DSmT.

DST (Dempster, 1968; Shafer, 1976) uses a single level as apposed to the dual level approach of the TBM; it is equivalent to the credal level of the TBM. Belief is assigned in the same manner as the TBM — with the exception that no mass can be assigned to the empty set. The degree of belief and degree of plausibility are calculated in the same way as the TBM. Since there is no pignistic level, beliefs are not converted into probabilities, but the degree of belief and degree of plausibility provide a lower and upper bound of probability:

$$bel(A) \leq P(A) \leq pl(A), \tag{2.35}$$

where $A \in \Omega$. Traditionally, *bbas* are combined within DST using Dempster's rule of combination (Equation 2.31).

DSmT (Dezert and Smarandache, 2009) differs from DST and the TBM in that it is not limited to the power set. The power set is the set of mutually exclusive hypotheses; DSmT can also use the hyper-power set or the super-power set — allowing overlapping hypotheses. Three models exist within DSmT, the free DSm model, the hybrid DSm model, and Shafer's model. The free DSm model has no mutual exclusivity constraints. The hybrid DSm model allows constraints to be introduced into the free model to better model reality. Shafer's model is equivalent to the power set, where all hypothesis are mutually exclusive. A thorough introduction to DSmT can be found in (Dezert and Smarandache, 2009).

Whereas the power set contains only the elements of Ω and the union of the elements of Ω, the hyper-power set also contains the elements formed by the intersection of the elements of Ω. The super-power set contains all the elements that would be in the hyper-power set in addition to the complement of the elements of Ω. For example (adapted from Dezert and Smarandache (2009)), if $\Omega = \{A, B\}$, then the power set is

$$2^{\Omega} = \{\emptyset, A, B, A \cup B\},$$

the hyper-power set, D^{Ω}, is

$$D^{\Omega} = \{\emptyset, A, B, A \cup B, A \cap B\},$$

and the super-power set, S^{Ω}, is

$$S^{\Omega} = \{\emptyset, A, B, A \cup B, A \cap B, \bar{A}, \bar{B}\}.$$

Both the hyper-powerset and the super-powerset allow the use of hypotheses that are not

mutually exclusive. The super-powerset is equivalent to the powerset of a larger frame of discernment. It should be noted that as $|2^{\Omega}| \leq |D^{\Omega}| \leq |S^{\Omega}|$, the use of the free DSm or hybrid DSm models is likely to increase computational complexity — particularly when combining belief assignments. The PCR rules were designed for use in DSmT; PCR5, discussed in Section 2.2.2.4, is used with the TBM in Chapters 5 and 6.

Both DST and DSmT have been discussed in this section. Due to a lack of an equivalent to the pignistic level, DST is not a suitable alternative to the TBM for the research of this thesis — probabilities are used as part of the tracking and classification process. The DSmT could be used as an alternative to the TBM as it contains all the features of the TBM. It is not necessary to replace the TBM with DSmT for the research presented in this thesis as the target classes are clearly defined and mutually exclusive. It is possible to envisage the use of DSmT instead of TBM in more realistic scenarios — but $|D^{\Omega}|$ or $|S^{\Omega}|$ will become too large at a much faster rate than that of $|2^{\Omega}|$ as $|\Omega|$ increases.

2.3 Information Theory

Information Theory can be used to quantify uncertainty of events, and in compression to minimise communication costs (Cover and Thomas, 1991, chap. 1). This section covers *entropy* and *mutual information*. Mutual information is used within the sensor selection algorithm that is utilised throughout this thesis. Only a basic understanding of the concepts of information theory is required for thesis.

2.3.1 Entropy

In communications, if events have different probabilities then on average, it may be possible to transmit less bits using variable length encoding. Entropy tells us on average, how many bits are required to transmit which event has occurred. Entropy can be calculated as follows (Cover and Thomas, 1991, chap. 2):

$$H(X) = -\sum_{x \in \mathcal{X}} p(x) \log p(x) \tag{2.36}$$

$$H(Y|X) = -\sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log p(y|x) \tag{2.37}$$

$$H(X,Y) = H(X) + H(Y|X) \tag{2.38}$$

Where x is an event in the sample space $\mathcal{X}$, y is an event in the sample space $\mathcal{Y}$, and p(x) is the probability of event x occurring.

Entropy is used in *Information-Driven Sensor Querying (IDSQ)* (Chu et al., 2002) and in Zhao et al. (2002) to optimise sensor selection. At each time step, a sensor is chosen to update the current belief of the target; Entropy is used to calculate, without knowing each sensor's measurement, which sensor will give the best *information gain* — the best reduction in error of the current belief of the target (The best improvement in *expected posterior belief*).

2.3.2 Mutual Information

Mutual information is a measure of how much the knowledge of a random variable reduces the uncertainty of another random variable (Cover and Thomas, 1991, chap. 2), it can be calculated as follows (Cover and Thomas, 1991, chap. 2):

$$I(X;Y) = H(X) - H(X|Y) \tag{2.39}$$

$$I(X;Y|Z) = H(X|Z) - H(X|Y,Z) \tag{2.40}$$

Ertin et al. (2003) builds upon IDSQ by using mutual information to choose which sensor's measurement will be used to condition the current estimate. It is shown that the *expected posterior uncertainty* is a better utility function than expected posterior belief — conditional entropy is used to quantify expected posterior uncertainty. This is used in Williams et al. (2007) to select the best leader node.

2.4 Sensor Management

WSNs must be managed in order to maximise their potential; limited power and network bandwidth typically make it infeasible to obtain measurements from all sensors in a WSN at all times — although this would be the optimal strategy if there were no constraints on resources.

Three sensor management strategies are discussed in this section: two that are *myopic* — they only plan for the next time step (Minimum expected distance and maximising mutual information); and one that plans over a rolling time horizon (Dynamic programming approach). The two myopic strategies select a single sensor, or leader node to minimise a cost function for each time step. The non-myopic strategy uses constrained dynamic programming to select both a leader node and a subset of sensors for each time step of the planning horizon.

2.4.1 Minimising expected distance

A simple and intuitive method for selecting a leader node for each time step, $k = \{1,2,\ldots,N_t\}$ (where N_t is the number of time steps), is to select the sensor, s_k, which will be closest to the expected position of the target:

$$s_k = \arg\min_{s \in \mathcal{S}} \mathrm{E}\left(||\mathbf{L}\mathbf{x} - \mathbf{y}^s||_2^2\right), \tag{2.41}$$

where $\mathcal{S} = \{1,2,\ldots,N_s\}$ is the set of possible sensors, N_s is the number of sensors, $\mathbf{L}$ is the matrix required to extract the position of the target from its state vector, $\mathbf{y}^s$ is the position of sensor s, and $||.||_2^2$ is the square distance squared.

We utilise range sensors; in order to achieve the highest SNR it is best to take measurements using nodes that are close to our target. Although this strategy achieves this, it does not take into account the over-loading of sensors when they are too close to the target. It also does not consider uncertainty — it may be wiser to use a sensor slightly further away if it gives a much larger reduction in the uncertainty of target location.

2.4.2 Maximising mutual information

Another myopic strategy for choosing a leader node is to select the sensor that will minimise the expected posterior entropy of the target state; this is the same as maximising mutual information (Ertin et al., 2003). the leader node for time step k can be selected as follows (Williams et al., 2007):

$$s_k = \arg\max_{s \in \mathcal{S}} I\left(\mathbf{x}_k; \mathbf{z}_k^s | \mathbf{z}_{0:k-1}\right), \tag{2.42}$$

where $\mathbf{z}_k^s$ is an observation taken from sensor s at time k and mutual information is calculated as:

$$I\left(\mathbf{x}_k; \mathbf{z}_k^s | \mathbf{z}_{0:k-1}\right) = H\left(\mathbf{z}_k^s | \mathbf{z}_{0:k-1}\right) - H\left(\mathbf{z}_k^s | \mathbf{x}_k\right), \tag{2.43}$$

and $H(.)$ is entropy.

2.4.3 Dynamic programming approach

The above two sensor management strategies only pick one node for a single time step. Williams et al. (2007), which extends previous work (Williams et al., 2005a,b), present a strategy for simultaneously selecting both a leader node and a subset of sensors for each time step of the planning horizon. Williams et al. tackle the trade-off between the quality of information obtained from a WSN and the energy costs of obtaining the

information. Dynamic programming is used optimise either energy usage or the quality of information subject to a constraint on the other quantity; in this thesis, we only utilise former of the two. The only energy cost considered is communications costs due to the fact that the cost of communications is the dominating factor of energy usage in a node (Pottie and Kaiser, 2000). A number of techniques are used to greatly reduce computational complexity allowing planning to be achieved within a reasonable time frame.

2.5 Conclusions

This chapter has given a brief overview of tracking, classification, information theory, and sensor management. Tracking has focussed on particle filtering which is used to estimate the kinematic state of a target using noisy observations. Classification has focussed on the TBM, which is an alternative approach to Bayesian probability theory that deals with uncertainty and ignorance in an intuitive manner. Information theory has been briefly discussed as mutual information is a key part of sensor management. Three sensor management approaches have been described, of which the dynamic programming approach is the most sophisticated.

Particle filtering, the TBM, and the dynamic programming approach to sensor management are combined in Chapter 4 to perform joint tracking and classification with a WSN. This is then extended in Chapter 5 to use terrain information and to perform in more realistic scenarios. All of this is achieved within the framework presented in Chapter 3.

Chapter 3

Executive Summary of Framework for JTAC with a WSN

Chapters 4 and 5 of thesis combine and extend an existing WSN tracking approach (Williams et al., 2007), and a JTAC approach that is not designed for WSNs (Powell et al., 2006) — this is accomplished within a framework that is presented in this chapter. The use of this framework to aid sensor deployment planning is then demonstrated in Chapter 6. A framework presents an attractive option for implementing these WSN tracking and JTAC approaches in a modular, extensible manner that facilitates code reuse and reduces development time. It is designed for WSN JTAC single target scenarios — although it is not necessarily limited to this. The framework is used to analyse algorithms within a simulated environment.

All of the experiments conducted for this thesis have been carried out within the framework presented in this chapter. A modular framework is used so as not to be limited to a particular planning algorithm, estimation algorithm, or scenario. The framework can be split into two major parts: the planning algorithm, and the estimation algorithm. An overview of the framework is given in this chapter and can be seen graphically in Figure 3.1.

Williams et al. (2007) show the use of their approach in different configurations and compare the results with existing strategies for sensor selection — this could be accomplished as a framework in which different sensor selection strategies can be combined with a target tracking algorithm.

The framework presented in this chapter can be seen as a more general framework in which the approaches of both Williams et al. (2007) and Powell et al. (2006) can be realised. The approaches presented and compared in Williams et al. (2007) can be reproduced by using using the appropriate planning algorithm to select sensors and by using a particle filter to provide tracking-only estimation, i.e. classification is not

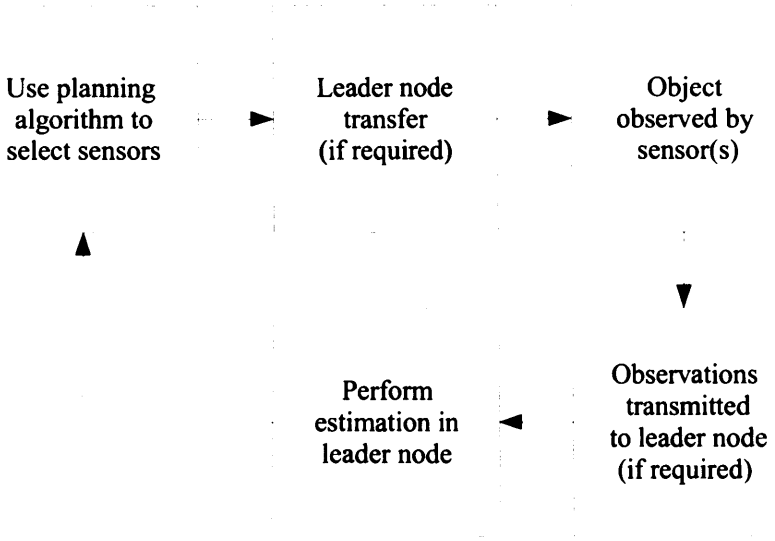


Figure 3.1: Overview of data flow within our framework

required. Even though the framework is designed for WSN applications, it can also be used to reproduce the JTAC approach of Powell et al. (2006) — this would require a single sensor and a simple planing algorithm that uses the same sensor at every time step.

It should also be possible to recreate or develop other approaches for tracking or JTAC within this framework. This framework could be ideally suited to enabling the rapid development of other similar WSN tracking and JTAC algorithms — including those that use different algorithms for estimation, such as a Kalman filter,

3.1 High-level Configurations

Three high-level configurations of the framework are used: tracking without classification, JTAC, and an improved JTAC algorithm that utilises terrain information. All three configurations use a particle filter for tracking. Tracking without classification is equivalent to the algorithm presented by Williams et al. (2007) (Discussed in Section 2.4.3). The former of the two JTAC configurations, discussed in Chapter 4, uses the TBM for classification in a similar way to how it is used by Powell et al. (2006). An improved approach to JTAC, presented in Chapter 5, builds upon the work in Chapter 4 — one of the ways it does this is by including terrain information in the classification process. Figure 3.2 shows how the similarities and differences between the high-level configurations. A high-level configuration will be customised for a particular scenario by changing various parameters.

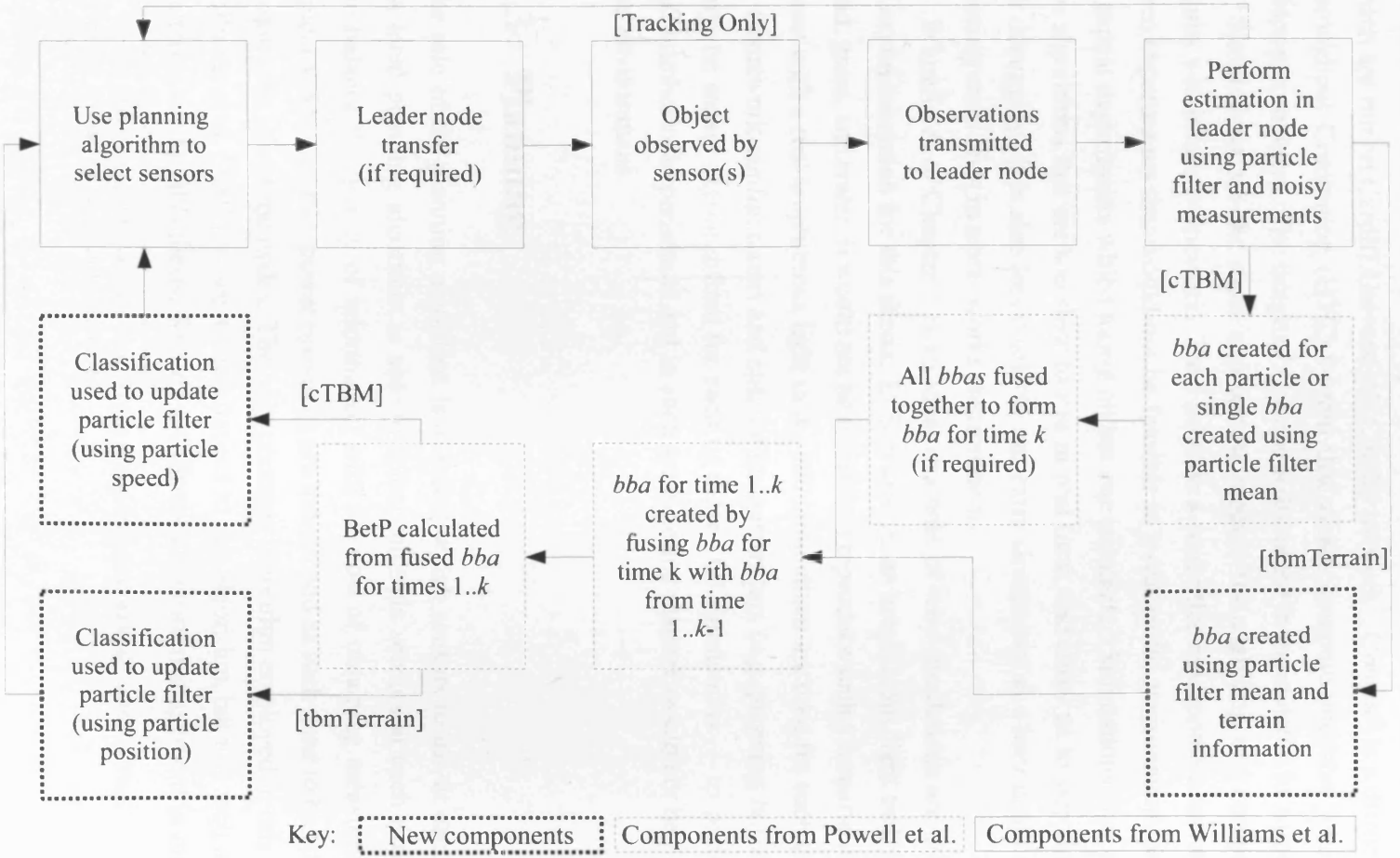


Figure 3.2: Comparison between high-level configurations of our framework. [Tracking Only] denotes tracking without classification, [cTBM] denotes the JTAC algorithm of Chapter 4, and [tbmTerrain] denotes the improved JTAC algorithm of Chapter 5.

3.2 Simulations

Simulations are used for all the experiments carried out for this thesis. The majority of which are run on Cardiff University's Condor network[1]. Condor[2] is a distributed High-Throughput Computing (HTC) system that allows compute-intensive jobs to on idle desktop computers. The usage of Condor is discussed in more detail in Appendix A.

Simulations are the most appropriate method for analysing and comparing algorithms within our framework. They provide a controlled environment in which to perform experiments that would not be feasible in a real world environment, and allow us to repeat experiments whilst fixing all but one parameter. Simulations also allows us to run algorithms that are too slow to run in real time, and allow us to step through code for debugging. It is also less costly to run extra simulations at a later date compared to running extra tests in a real world environment.

Scenario A of Chapter 5 is an ideal example of why simulations are the best form of experimentation for this thesis. In Scenario A, an amphibious light tank travels over road, grass, and water. It would not be feasible to reproduce such a scenario for an experiment with a real amphibious light tank. 100 simulations are used for each combination of framework configuration and simulation parameters (e.g. planning horizon length); only the sensor layout differs for each of these 100 simulations — to perform such a vast number of experiments and in such a controlled manner would not be possible in a real environment.

3.3 Planning

The role of the planning algorithm is to select which sensors to use at each time step. An ideal planning algorithm is able to select multiple sensors at each time step, and can balance the quality of information with the cost of obtaining measurements; in a typical WSN scenario, power resources are limited and as such need to be considered to prolong the life of the nodes. The only planning algorithm employed in this thesis is by Williams et al. (2007) — we are not limited to this algorithm, but it is well suited to the scenarios used in this thesis. Section 2.4 discusses sensor management in more detail.

[1] http://www.cardiff.ac.uk/arcca/services/condor/arcca-condor.html
[2] http://www.cs.wisc.edu/condor/

3.4 Estimation

The estimation part of the framework is for target tracking and classification. All of the three high-level configurations use a particle filter for tracking, although the framework is not limited to using this. Both JTAC configurations use the TBM for classification.

At each time step, the estimate of the target's kinematic state is updated using distance measurements from the WSN nodes selected by the planning algorithm. This estimate is then used to create conditional belief masses in the credal level of the TBM. The classification from the pignistic transform is then used to update the particle filter distribution. This feedback loop can be seen in Figure 3.2.

The same general particle filter configuration is used throughout this thesis. Target dynamics are modelled with a linear state update — although both constant acceleration and white-noise acceleration target dynamics can be accounted for by changing the magnitude of the process noise. A non-linear measurement model is used, which is more realistic than a linear sensing model. The particle filter configuration originates from previous work by Williams et al. (2007). Simulations in our work have shown the need for a particle filter, rather than an EKF, as the PDF of the target can become significantly non-Gaussian — particularly when a target approaches a actively sensing node.

3.4.1 Joint Tracking and Classification

The JTAC approach of Chapter 4 combines and extends previous works by Powell et al. (2006) and Williams et al. (2007). The TBM is used to perform classification, at each time step a *bba* is created based on the estimated target state. This *bba* is either created directly from the mean estimate, or is created by generating a *bba* for each particle and fusing them together. The *bba* for the current time step is fused with the existing *bba* (from previous time steps) using closed world conjunctive combination; this forms the new up to date *bba*. The pignistic transform is then used to calculate class probabilities, which are then used to update the particles. This feedback loop between tracking and classification seeks to not only improve JTAC, but also sensor selection as this is affected by the uncertainty of the target state.

The plausibilities used to create the *bba* or *bbas* at each time step are conditional on the estimated target speed. This approach is sufficient for constant velocity targets but does not work well with the scenarios of Chapters 5 and 6 as targets have non-constant acceleration. A potential improvement to this would be to create plausibilities conditional on target acceleration, but unfortunately the estimate of this is typically too noisy.

3.4.2 Using Terrain Information to Improve Joint Tracking and Classification

An improved approach to calculating conditional plausibility is used in Chapter 5; plausibility is still conditional on the target's speed, but terrain information is used to weight a combination of plausibilities. The different plausibilities reflect the likely speeds of a target class travelling over different terrain classes. A terrain class is a group of terrain types that have a similar hindrance to the motion of a target class. Three terrain classes are used — 'Go', 'Slow Go', and 'No Go'. For example, the only terrain type for the 'Go' target class 'Car' is 'Road'. 'No Go' terrain types for the car include 'Water' and 'Marsh'. This improvement makes it possible to track targets that travel over different terrain types and exhibit more complex trajectories — usually as a result of the varying terrain. The similarities and differences between the approaches of Chapters 4 and 5 can be seen in Figure 3.2.

3.5 Sensor Deployment Planning

As demonstrated in Chapter 6, the framework can also be used as a tool to aid sensor deployment planning. Simulations can be used within our framework to investigate how sensor deployment affects the performance of JTAC approaches. This alternative use of the framework can be used to help decide where best to deploy sensors for a given scenario or set of scenarios, which results in an improved performance. This tool could be used prior to sensor deployment in real world scenarios to help provide a low cost means of selecting a good sensor layout.

3.6 Conclusions

This chapter has introduced a framework for JTAC and summarised its use in this thesis. It is a modular and extensible framework to assist both algorithm development and sensor deployment planning. Simulations are used instead of real world experiments as they are more suitable for the research carried out within this thesis.

Three main high-level configurations of the framework are used: tracking without classification, JTAC, and an improved JTAC approach that utilises terrain information. Tracking is performed using a particle filter and noisy range measurements. Classification takes place with the TBM — a closed world, discrete frame of discernment is used, *bbas* are conditional upon the speed of the target. The same planning algorithm,

by Williams et al. (2007), is used by all the configurations due to its ability to select multiple sensors and to plan for more than just the next time step.

The next three chapters will show the use of the framework to both analyse the performance of these high-level configurations in different scenarios, and to demonstrate the use of this framework to aid sensor deployment planning. Chapters 4 and 5 discuss the high-level configurations in more detail. The use of this framework in planning sensor deployment is discussed in Chapter 6.

Chapter 4

Joint Tracking and Classification with Wireless Sensor Networks

The joint tracking and classification of targets can be important as there may only be a need to track certain types of objects. The identity of a target can also be used to update the belief that is held about a target's kinematic state. This chapter presents a novel method to jointly track and classify a target with a WSN, whilst also planning the usage of sensors in a such a way that balances the cost of communications with the quality of information obtained by the sensors. We combine and extend work by Powell et al. (2006) and Williams et al. (2007). Tracking and data fusion is performed with a particle filter (Arulampalam et al., 2002). The cTBM (Smets, 2005) is used for classification conditional on target speed. A dynamic programming approach by Williams et al. (2007) is used for sensor management — although we are not limited a single sensor management strategy. JTAC takes place within the framework presented in Chapter 3. This chapter is an expanded version of Roberts and Marshall (2008).

Existing Bayesian approaches to joint tracking a classification include work by Gordon et al. (2002), and Challa and Pulford (2001) — which highlights the inter-dependence between target class and target state. Powell et al. (2006) perform JTAC of airborne targets using a particle filter and the cTBM. The TBM has been used for fusing sensor data in autonomous land vehicle positioning (Caron et al., 2005); this approach may not be suitable for tracking within the context of WSNs as it is common not to have such a rich set of sensor types, as well as not having sensors on board the target. Work by Smets and Ristic (2004) performs JTAC by re-deriving the Kalman filter within the TBM framework, however we use a particle filter as we do not wish to limit ourselves to a linear measurement model.

Section 4.1 provides an brief overview of the two systems that are extended. The problem formulation is presented in Section 4.2. A new method for providing feedback

from the cTBM to a particle filter is presented along with details of how the systems by Williams et al. and Powell et al. are implemented in Section 4.3. An analysis of the new system and a performance comparison with the system by Williams et al. is provided in Section 4.4. Finally, conclusions and future work are discussed in Section 4.5.

4.1 Background

We broadly follow the notation of Williams et al. (2007) and Powell et al. (2006) — both of which are briefly described in this section. For more details of the above approaches, it is recommended that their respective literature is consulted. A brief overview of the cTBM can be found in Section 2.2.2.3 and sensor management approach of Williams et al. in Section 2.4.3.

4.1.1 Joint Tracking and Classification

A system presented by Powell et al. (2006) integrates the cTBM and a particle filter to perform JTAC of aircraft. Classification is achieved by using *bbms* conditional on target speed; the conditional plausibility function is continuous (due to target speed), but the possible outcomes are discrete target types. The approach presented results in a more robust classification than a Bayesian classifier. All the *bbms* (and hence each *bba*) are conditional on speed.

At each time step, each particle is used to create a *bba*. Discounting is then used to weight each *bba* with respect to its particle weight; this is required as it enables the cTBM to deal with uncertainty. The *bbas* produced from particles with a larger uncertainty will have less of an impact during classification. Discounted masses, m^{α}, are calculated using Equation 2.34, where the discounting factor is equal to the weight of the particle that is used to create the *bba*, $\alpha = w_k^i$.

Discounted *bbas* for the current time step are fused, the resulting *bba* is then fused with the existing *bba* created from previous time steps. Each particle is used in the classification process instead of using the mean of the particles, this retains data that could otherwise be lost (Powell et al., 2006). The *bbas* are fused recursively using a closed world. Belief masses are fused using Dempster's rule of combination (Equation 2.31). An open world cannot be used as the mass assigned to the empty set would quickly converge to 1. This is because each particle is different — they do not perfectly agree with each other, this is especially the case with outlying particles.

When a closed world is used with Dempster's rule of combination, fusing *bbas* can lead to a convergence on a singleton set. If this happens, the cTBM will be unable

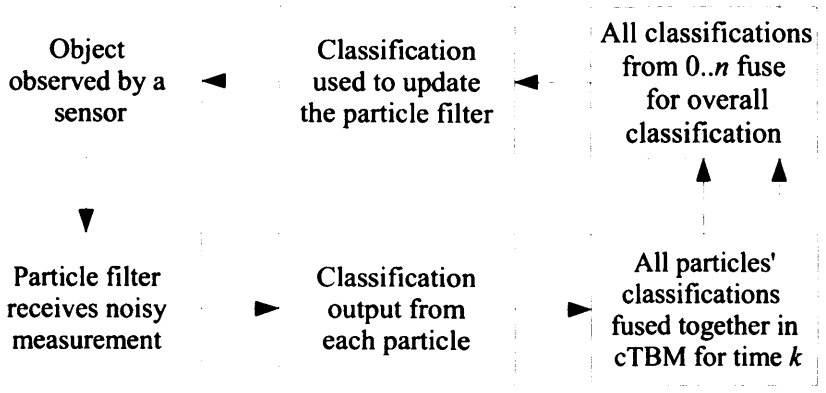


Figure 4.1: Data flow of Powell et al.

to assign belief to another set at a later time, which could result in the cTBM falsely classifying a target. Convergence protection, presented by Powell et al. (2006), prevents this — an upper limit of 0.99 is placed on the mass that can be assigned to a singleton set. If this limit is reached, the excess is removed from the singleton set and assigned to $m(\Omega)$.

After fusing the *bbas*, the pignistic transform (Equation 2.27) is used to create probability functions for classification. This classification is used to condition the particle filter. In the next time step, a new measurement is taken and used to update the particle filter, this creates a feedback loop — Figure 4.1, from Powell et al. (2006), provides an overview of this loop. In Section 4.3 we will show how the feedback loop in our system differs from this.

Feedback from the cTBM to the particle filter updates the particle filter to reflect the output from the cTBM. Unfortunately Powell et al. (2006) do not specify how this feedback is implemented — this makes it impossible for us to replicate this part of their system. Section 4.3 presents a new method for this feedback.

4.1.2 Tracking with Sensor Management

We use the sensor management approach of Williams et al. (2007) which is briefly discussed in Section 2.4.3. Sensor management takes place within the context of tracking a single target. The tracking aspect of the work by Williams et al. is discussed here.

Williams et al. use SIS with resampling at each time step for tracking. Sensor measurements are fused within the particle filter. The pdf of the target's kinematic state, $\mathbf{x}_k$, conditional on measurements received up to and including time k, $\mathbf{z}_{1:k}$, is approximated

by

$$p(\mathbf{x}_k|\mathbf{z}_{1:k}) \approx \sum_{i=1}^{N_p} w_k^i \delta(\mathbf{x}_k - \mathbf{x}_k^i), \tag{4.1}$$

where N_p is the number of particles, w_k^i is the weight of i^{th} particle $\mathbf{x}_k^i$, at time k, and $\delta(.)$ is the Dirac delta function. The same distribution is calculated for the next time step by sampling from the proposal distribution:

$$q\left(\mathbf{x}_{k+1}|\mathbf{x}_k^i, \mathbf{z}_{k+1}\right) = \mathcal{N}\left(\mathbf{x}_{k+1}; \hat{\mathbf{x}}_{k+1}^i, \mathbf{P}_{k+1}^i\right) \tag{4.2}$$

where

$$\begin{aligned}
\mathcal{N}(\mathbf{x};\mu,\mathbf{P}) &= |2\pi\mathbf{P}|\, e^{\frac{1}{2}(\mathbf{x}-\mu)^T\mathbf{P}^{-1}(\mathbf{x}-\mu)} && (4.3)\\
\hat{\mathbf{x}}_{k+1}^i &= \mathbf{F}\mathbf{x}_k^i + \mathbf{K}_{k+1}^i\left[\mathbf{z}_{k+1} - h(\mathbf{F}\mathbf{x}_k^i, s)\right] && (4.4)\\
\mathbf{P}_{k+1}^i &= \mathbf{Q} - \mathbf{K}_{k+1}^i \mathbf{H}^s\left(\mathbf{F}\mathbf{x}_k^i\right)\mathbf{Q} && (4.5)\\
\mathbf{K}_{k+1}^i &= \mathbf{Q}\left\{\mathbf{H}^s\left(\mathbf{F}\mathbf{x}_k^i\right)\right\}^T\left[\mathbf{H}^s\left(\mathbf{F}\mathbf{x}_k^i\right)\mathbf{Q}\left\{\mathbf{H}^s\left(\mathbf{F}\mathbf{x}_k^i\right)\right\}^T + \mathbf{R}^s\right]^{-1}. && (4.6)
\end{aligned}$$

$\mathbf{z}_{k+1}$ is the observation at time $k+1$, $h(\mathbf{F}\mathbf{x}_k^i, s)$ is the measurement function for the subset of sensors $s \subseteq S = \{1,\ldots,N_s\}$ (where N_s is the number of sensors) when observing a target state $\mathbf{F}\mathbf{x}_k^i$, $\mathbf{Q}$ is the covariance matrix for the process noise of the target state, $\mathbf{F}$ is the matrix that projects the object dynamics from step k to $k+1$, $\mathbf{R}^s$ is the covariance of the measurement noise for sensors s, and $\mathbf{H}^s\left(\mathbf{F}\mathbf{x}_k^i\right)$ is a linearisation of $h(\mathbf{F}\mathbf{x}_k^i, s)$ about a nominal point $\mathbf{F}\mathbf{x}_k^i$. $h(\cdot, s)$ is a vector-valued function of length equal to $|s|$. $\mathbf{z}_{k+1}$ is a vector of length also equal to $|s|$. Each particle is conditioned on all of the measurements received at the current time step. $\mathbf{F}$, $\mathbf{Q}$, $h(\cdot)$, $\mathbf{H}^s(\cdot)$, and $\mathbf{R}^s$ are defined in Section 4.2.

The weight of particle $\mathbf{x}_{k+1}^i$ is

$$w_{k+1}^i = c w_k^i \frac{p\left(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}^i\right) p\left(\mathbf{x}_{k+1}^i|\mathbf{x}_k^i\right)}{q\left(\mathbf{x}_{k+1}^i|\mathbf{x}_k^i, \mathbf{z}_{k+1}\right)}, \tag{4.7}$$

where

$$p\left(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}^i\right) = \mathcal{N}\left(\mathbf{z}_{k+1}; h(\mathbf{x}_{k+1}^i, s), \mathbf{R}^s\right), \tag{4.8}$$

and c is a normalisation constant such that $\sum_{i=1}^{N_p} w_{k+1}^i = 1$. In Section 4.3 we modify Equation 4.7 to take into account the belief held by the cTBM.

By moment-matching the pdf to a Gaussian distribution, the mean and covariance

are calculated as

$$\mu_k = \sum_{i=1}^{N_p} w_k^i \mathbf{x}_k^i \quad \text{and} \quad \mathbf{P}_k = \sum_{i=1}^{N_p} w_k^i \left(\mathbf{x}_k^i - \mu_k\right)\left(\mathbf{x}_k^i - \mu_k\right)^T. \tag{4.9}$$

4.2 Problem Formulation

We use the same object dynamics, sensor models, and communication models as Williams et al. (2007), which are briefly outlined in this section. As with Williams et al., our approach is not limited to this problem, it is simply used to provide a specific example with which it was tested. For more information about target dynamics models, see Li and Jilkov (2003).

The target state at time k is $\mathbf{x}_k = [pos_x\, vel_x\, pos_y\, vel_y]^T$, where pos_x and pos_y are the x and y coordinates of the target, respectively, and vel_x and vel_y is the velocity of the target in the x and y axes, respectively. The dynamics of the target state are updated by

$$\mathbf{x}_{k+1} = \mathbf{F}\mathbf{x}_k + \mathbf{w}_k, \tag{4.10}$$

where

$$\mathbf{F} = \begin{bmatrix} 1 & T & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & T \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.11}$$

and $\mathbf{w}_k \sim \mathcal{N}(\mathbf{w}_k; 0, \mathbf{Q})$, and T is the sampling interval. The covariance of the process noise is calculated as follows:

$$\mathbf{Q} = q \begin{bmatrix} \frac{T^3}{3} & \frac{T^2}{2} & 0 & 0 \\ \frac{T^2}{2} & T & 0 & 0 \\ 0 & 0 & \frac{T^3}{3} & \frac{T^2}{2} \\ 0 & 0 & \frac{T^2}{2} & T \end{bmatrix}. \tag{4.12}$$

A non-linear sensor model is used, the measurement obtained by sensors s_k at time k is calculated by

$$\mathbf{z}_k^s = h(\mathbf{x}_k, s) + \mathbf{v}_k^s \tag{4.13}$$

where $\mathbf{v}_k^s \sim \mathcal{N}\left(\mathbf{v}_k^s; 0, \mathbf{R}^s\right)$, it is independent for each s and independent of $\mathbf{w}_k$, $\mathbf{R}^s$ is the measurement noise covariance matrix for sensors s, and

$$h(\mathbf{x}_k, s) = \frac{a}{||\mathbf{L}\mathbf{x}_k - \mathbf{y}^s||_2^2 + b}. \tag{4.14}$$

a and b are used to model the SNR ratio of sensors s. The measurement from each sensor has additive Gaussian noise with variance R.

A linearisation of the above equation about a nominal point, $\mathbf{x}^0$, is

$$\mathbf{H}^s(\mathbf{x}^0) = \frac{-2a}{(||\mathbf{L}\mathbf{x}_k - \mathbf{y}^s||_2^2 + b)^2}(\mathbf{L}\mathbf{x}^0 - \mathbf{y}^s)^T\mathbf{L}, \tag{4.15}$$

it is used in the particle filter, and also used by Williams et al. to reduce the complexity of the planning stage.

Any sensor can communicate with any other sensor, where the cost per bit of direct communication between nodes i and j is

$$\tilde{C} \propto ||\mathbf{y}^i - \mathbf{y}^j||_2^2. \tag{4.16}$$

Multi-hop communication is used to minimise communication costs. The total communication cost from node i to j, C_{ij}, is the sum of the direct communications costs between between the nodes along the shortest path, $\{i_0, i_1, \ldots, i_{n_{ij}}\}$, where $i_0 = i$, $i_{n_{ij}} = j$, and

$$C_{ij} = \sum_{k=1}^{n_{ij}} \tilde{C}_{i_{k-1}i_k} \tag{4.17}$$

Two different types of communications occur within the WSN — the transmission of measurements and of the probabilistic model. Measurements are sent from a subset of nodes to the current leader node for data fusion within the particle filter. When a new leader node is chosen, the probabilistic model is transferred from the current leader node to the new one. It assumed that the costs of these are fixed, the number of bits in a measurement is B_m, and the number of bits in the probabilistic model is B_p, therefore the cost of sending a measurement and the cost of sending the probabilistic model between nodes i and j are $B_m C_{ij}$ and $B_p C_{ij}$ respectively.

4.3 Integration

This section describes how we integrate and extend two existing systems: one for JTAC, and one for sensor management. Powell et al. (2006) is used for JTAC, and Williams et al. (2007) is used for sensor management. A feedback loop exists in both of the aforementioned systems, a particle filter is a common component of both. Our approach combines the systems in such a way that a single feedback loop contains all of the components of both systems; some of the components are modified to enable this, but

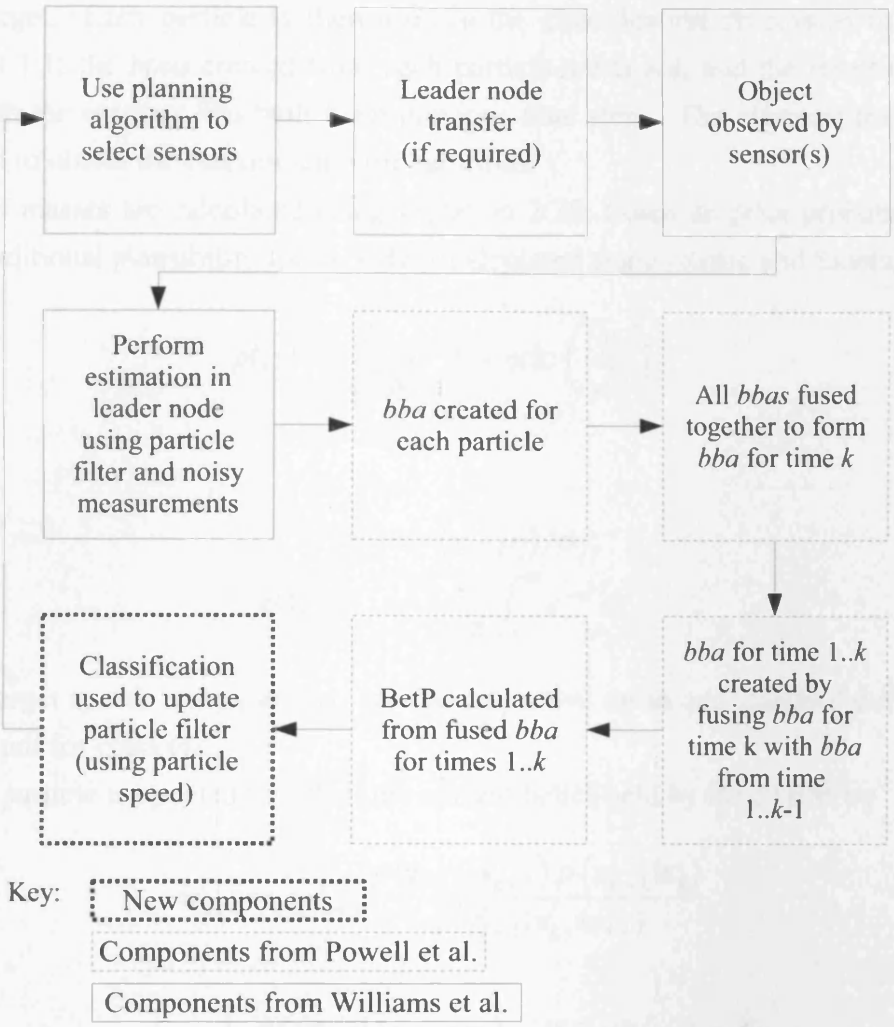


Figure 4.2: Our data flow

this does not change the function of each component. Figure 4.2 shows the feedback loop, highlighting how Powell et al.'s data flow differs from ours.

The use of Williams et al. for sensor management is not required for our approach to work. We could use a different strategy such as those outlined in Section 2.4.1 or 2.4.2; however, Williams et al. is more suitable as it seeks to achieve a balance between communication cost and the value of information obtained from sensors whilst being non-myopic, and is computed within an acceptable time.

Our approach starts by using the sensor selection algorithm for the current time step, if a new leader node is required, then the probabilistic model is transferred to the new leader. Each activated sensor (including the leader node) makes an observation which, if required, is transmitted to the leader node. The particle filter described in Section 2.4.3 is then used to fuse sensor measurements and update the pdf of the kinematic state

of the target. Each particle is then used in the classification process as outlined in Section 4.1.1; the *bbas* created from each particle are fused, and the resulting *bba* is fused with the existing *bba* built from previous time steps. The pignistic transform is then used to obtain the classification probabilities.

Belief masses are calculated using Equation 2.25; Gaussian prior probabilities are used, conditional plausibility for $\omega_j \in \Omega$ is calculated using (Ristic and Smets, 2004):

$$pl(y) = \frac{2y}{\sqrt{2\pi}} e^{-\frac{y^2}{2}} + \text{erfc}\left(\frac{y}{\sqrt{2}}\right), \tag{4.18}$$

where

$$y = (x - \mu_j)/\sigma_j, \tag{4.19}$$

$$\text{erfc}(s) = \frac{2}{\sqrt{\pi}} \int_s^\infty e^{-t^2} dt, \tag{4.20}$$

x is the target speed, and μ_j and σ_j are the respective mean and standard deviation of the prior pdf for class ω_j.

Each particle is updated to reflect the current belief held by the cTBM by

$$w_{k+1}^i = c\bar{w}_{k+1}^i \frac{p\left(\mathbf{z}_{k+1}|\mathbf{x}_{k+1}^i\right) p\left(\mathbf{x}_{k+1}^i|\mathbf{x}_k^i\right)}{q\left(\mathbf{x}_{k+1}^i|\mathbf{x}_k^i, \mathbf{z}_{k+1}\right)}, \tag{4.21}$$

where

$$\bar{w}_{k+1}^i = \begin{cases} \mathcal{N}\left(\mathbf{S}\left(\mathbf{x}_k^i\right); \mu_j, \gamma\sigma_j\right) & \text{if } BetP(\omega_j) > \beta; \\ 1 & \text{if } BetP(\omega_j) \leq \beta. \end{cases} \tag{4.22}$$

$\mathbf{S}\left(\mathbf{x}_k^i\right)$ extracts the speed from the state vector $\mathbf{x}_k^i$, the condition factor, γ, is used to reduce the impact of $\mathcal{N}$, and j is the defined such that:

$$\underset{j \in \Omega}{\arg\max}\, BetP(\omega_j). \tag{4.23}$$

Equation 4.21 is used to adjust the particle weights to reflect the current belief of the cTBM once it shows that the target classification has reached a classification threshold, β. Systematic resampling (See Section 2.1.2.3) is then performed with the particles and their adjusted weights.

4.4 Results

Monte Carlo trials were used to test the integration presented in Section 4.3. We compared our approach to that of Williams et al. (2007); communication constrained dynamic programming was used for both, with rolling horizon lengths $N = 5$, 10, and 25. 100 Monte Carlo trials were used for each combination of parameters. For example, if $N = 5$, 10, and 25, and $\gamma = 2$, 3, and 5, then $100 \times 3 \times 3 = 900$ Monte Carlo trials are required. We have used the same parameters for communication constrained sensor management as that of Williams et al. (2007).

The initial target state for each trial was $\mathbf{x}_1 = [0\,2\,0\,2]^T$, the sampling interval $T = 0.25$ seconds, and $q = 10^{-10}$. The simulations were limited to a 100 by 100 region and $N_t = 200$, if the target left the region or the current time step reached N_t then the simulation was stopped. The measurement model parameters were $a = 2000$, $b = 100$, and $R = 1$. The communication cost parameters were $B_p = 1$ and $B_m = 64$.

Each trial used a different layout of 20 randomly positioned sensors, and a different target trajectory. The same set of 100 sensor layouts were used for each set of 100 Monte Carlo trials. Sensors layouts were generated using a uniform distribution — layouts that did not appear to be uniformly distributed were discarded.

The cTBM is provided with the prior class probabilities shown in Figure 4.3a and Table 4.1. A classification threshold of $\beta = 0.8$, and a condition factor of $\gamma = 2$, 3, and 5 was used. In a typical Monte Carlo trial the target is classified correctly and with a sufficient probability within approximately 5 to 10 time steps (See Figure 4.3b), this will continue for the rest of the scenario (See Figure 4.5), resulting in Equation 4.21 having an impact on the particle distribution for most of each simulation.

All of the box plots shown in this thesis use the same notation — the red vertical line is the median of the distribution, the left and right edges of each box are the first and third quartiles, respectively. The whiskers, denoted by the black dashed line, extend to the lowest and highest non-outlier data points. Outliers, denoted by a '+', are less than $q_1 - 1.5(q_3 - q_1)$ or more than $q_3 + 1.5(q_3 - q_1)$, where q_1 and q_3 are the first and third quartiles, respectively.

Figure 4.4 provides a summary performance comparison of the framework with and without integrating the cTBM. Figures 4.5, 4.6, and 4.7 give a more detailed view of the classification performance, tracking performance, and communications costs respectively — each figure provides a detailed view by creating a box plot for each set 100 Monte Carlo trials. The overall performance of the framework with and without the cTBM are very similar. The Monte Carlo trials are clustered by the planning horizon

Target Class (ω_j)	Average Speed (μ_j)	Standard Deviation (σ_j)
A	2.8284	0.8
B	10	2
C	30	5

Table 4.1: The pdf priors for the cTBM

length with respect to accrued communication cost (See Figures 4.4 and 4.7) — this is an expected effect of using the sensor management algorithm by Williams et al. (2007).

The results show similar tracking performance regardless of whether classification is used. When classification is used, its performance is very good. Classification is limited to closed world scenarios, and due to the fusion of one *bba* for each particle, is computationally intensive.

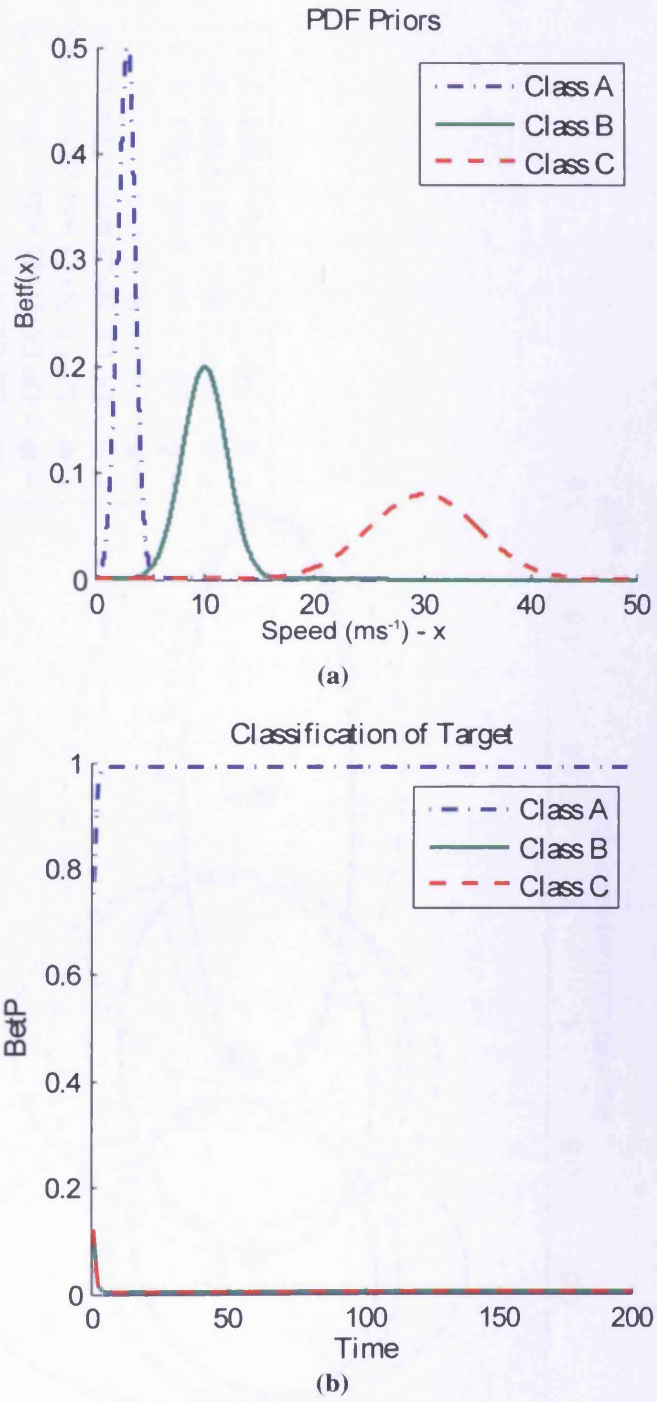


Figure 4.3: The pdf priors for the cTBM (a) and a typical classification output from the cTBM for a Monte Carlo trial (b).

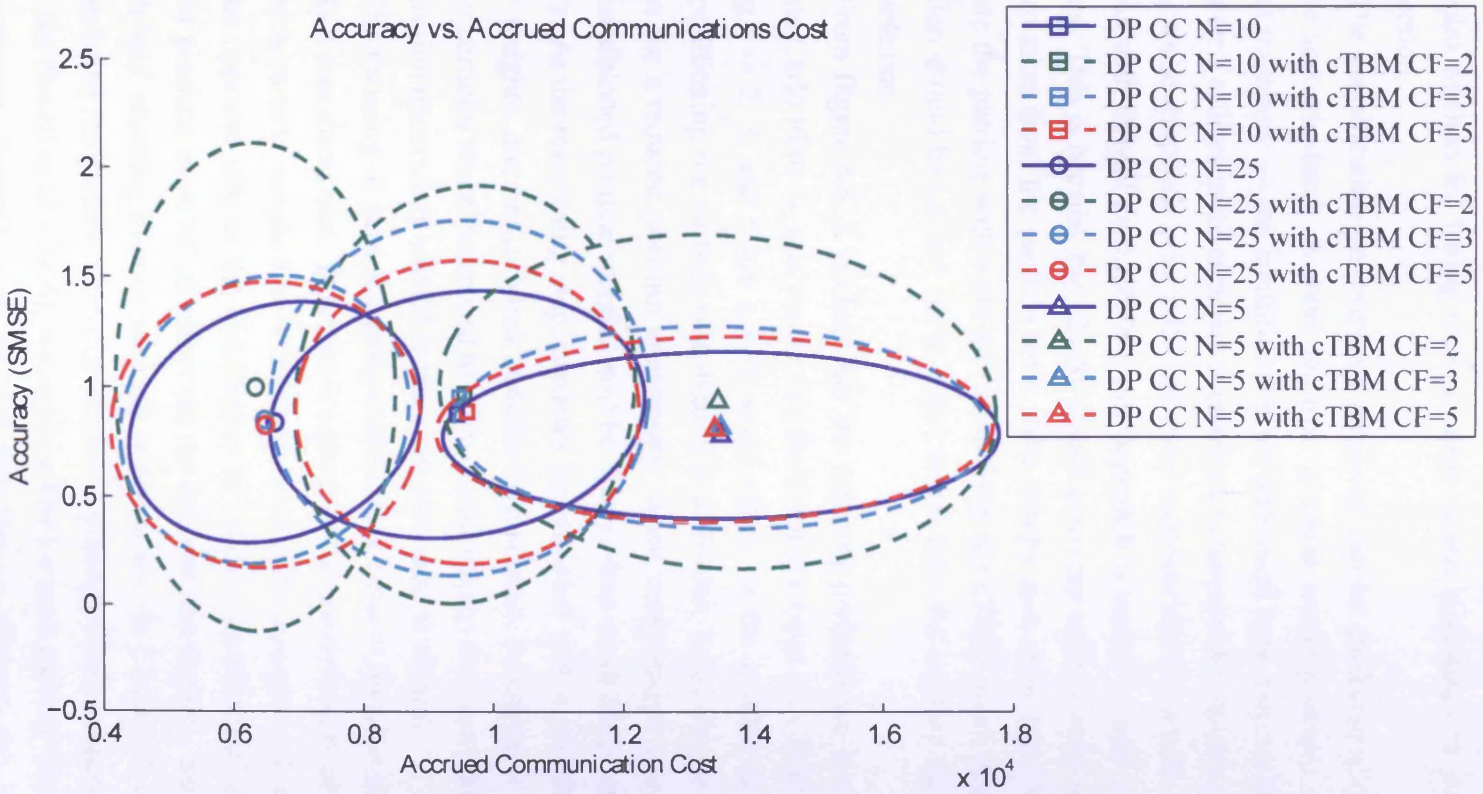


Figure 4.4: A comparison of William et al.'s system ('DP CC') and ours ('DP CC with cTBM') with planning horizon lengths $N = 5$, $N = 10$, and $N = 25$. For the cTBM, condition factors ('CF') of $\gamma = 2$, $\gamma = 3$, and $\gamma = 5$ are used. The ellipses show how the results vary across simulations, the centre of each ellipse is the mean, the edge of each ellipse is 1 standard deviation (Assuming a Gaussian distribution). This figure compares accrued communications cost with track accuracy (SMSE).

With the use of convergence protection in this scenario, the highest classification possible for class A (or any other singleton class) is $\frac{m(A)}{|A|} + \frac{m(\Omega)}{|\Omega|} = 0.99 + \frac{0.01}{3} = 0.99\dot{3}$ — the results shown in Figure 4.5 appear to approach this limit. In Chapter 5 a more complex method for fusing *bbms* is used, which eliminates the need for convergence protection.

The classification performance achieved can be attributed to the simplistic nature of the scenario used. A more realistic scenario would probably contain a non-linear target trajectory — the conditional prior pdfs used here may not be sufficient for this. Chapter 5 utilises more complex scenarios to compare the classification performance of using these simplistic prior pdfs and more sophisticated prior pdfs.

As with Powell et al. (2006), our approach is unable to deal with open world scenarios. This is because the cTBM would converge to the empty set due to conflicting information from the particle filter. The results also show that the method we use to update the particle with a classification from the cTBM is not optimal. Further investigation should be carried out to either better tune the existing method, or find a more suitable one.

From Figure 4.6, it is clear that the tracking performance is similar for all sets of Monte Carlo trials — this shows that the feedback from the cTBM to the particle filter using $\gamma = 2$, 3, and 5 has a very small effect on the particle distribution. The aim of conditioning the particle distribution is to further focus the particles to more likely states for a reduced position uncertainty; when compared to unconditioned particles, the conditioned particle weights will be larger when more likely and smaller when less likely. At the resampling step, particles are sampled with a probability proportional to their weights, and hence when conditioning is used, sampling is biased toward more likely particles when compared to when conditioning does not take place. A reduction in position uncertainty should result in an increased accuracy.

The focusing of particles using conditioning aims to increase accuracy, but unfortunately it introduces bias. This bias is a result of the uneven nature of particle distribution; for each particle weight that is changed, there is no guarantee of an equivalent particle on the opposite side of the distribution to create a symmetrical change. A smaller γ would produce more of an effect on the particle distribution, but would produce too much bias, resulting in poor tracking performance. In Chapter 5, a less biased method for updating the particle filter distribution is presented and compared to this method.

Like Powell et al. (2006), we create a *bba* for each particle which requires a significant amount of computation for each time step — although this provides a more rich classification, in scenarios such as this the mean of the particle distribution should be sufficient. Work presented in Chapter 5 creates a single *bba* for each time step for this

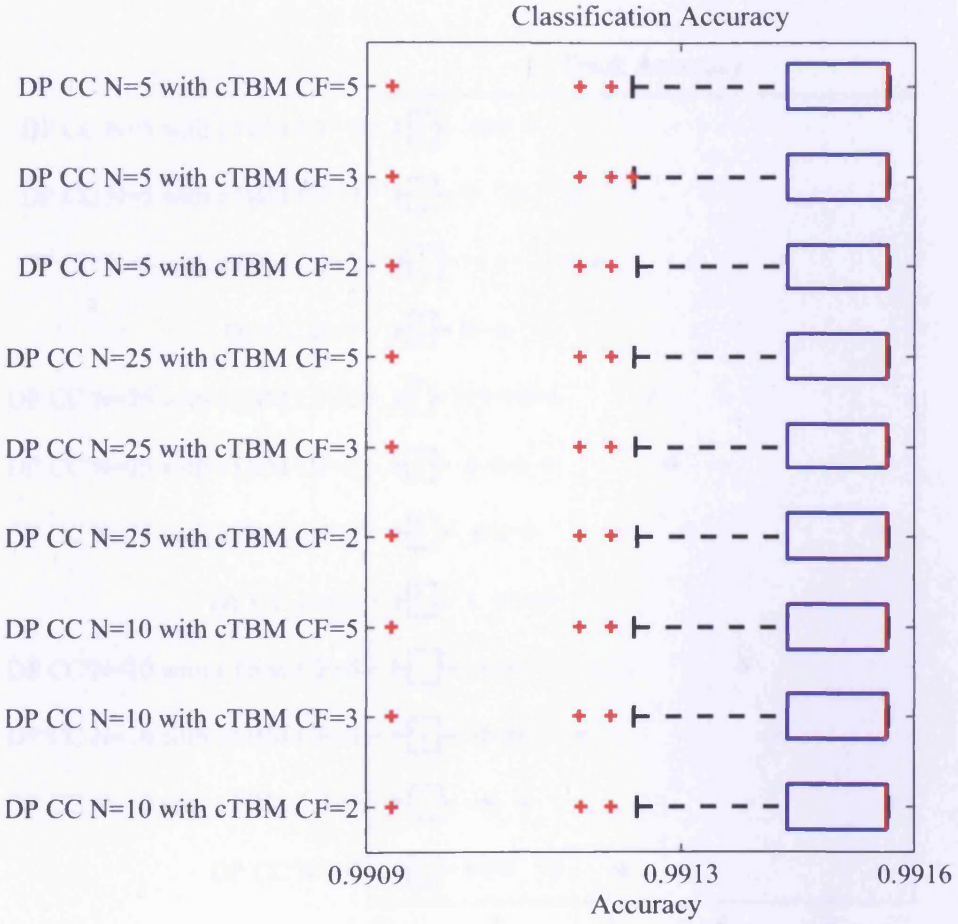


Figure 4.5: Classification accuracies for our framework. Each box plot is created from the mean value of $BetP(A)$ over time for each simulation. From the above box plots it is clear that for the majority of each Monte Carlo trial, the target is classified correctly, and above the threshold for particle conditioning, $\beta = 0.8$ — this results in conditioning taking place for most of each Monte Carlo trial.

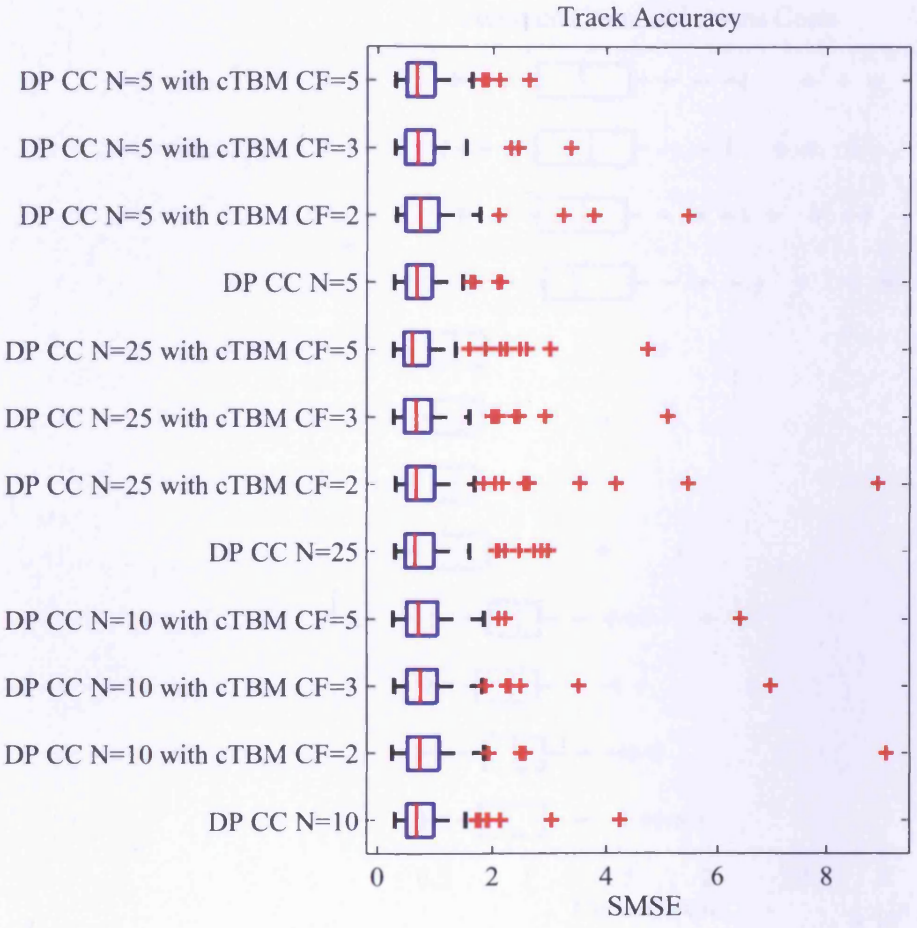


Figure 4.6: A comparison of track accuracies. Each box plot is created from a set of 100 Monte Carlo trials. The Sum of the Mean Squared Errors (SMSE) is used to compare tracking accuracy.

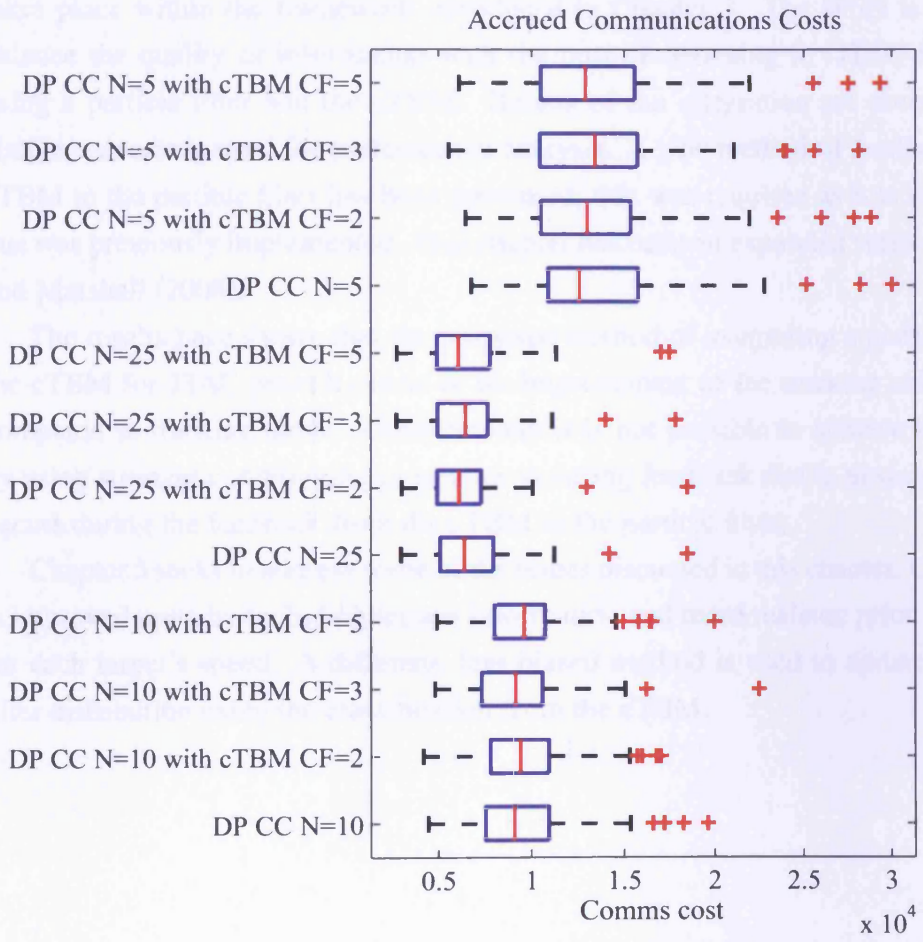


Figure 4.7: Accrued communications costs. Each box plot is created from a set of 100 Monte Carlo trials.

reason. It may also be possible to reduce the amount of computation required for fusing *bbas* by using methods discussed in Martin (2009).

4.5 Conclusions

In this chapter, we have integrated and extended work by Williams et al. (2007) and Powell et al. (2006) to achieve a novel JTAC algorithm for WSNs. This integration takes place within the framework introduced in Chapter 3. The WSN is managed to balance the quality of information with the cost of obtaining it. JTAC is performed using a particle filter and the cTBM. Details of the integration are presented, and a simple scenario is used for performance analysis. A new method of feedback from the cTBM to the particle filter has been presented; this was required as it is not clear how this was previously implemented. This chapter has been an expanded version of Roberts and Marshall (2008).

The results have shown that the proposed method of integrating a particle filter and the cTBM for JTAC provides little or no improvement to the tracking accuracy when compared to tracking alone. Unfortunately it is not possible to achieve better results by using stronger conditioning (a smaller γ) during feedback due to unwanted bias that occurs during the feedback from the cTBM to the particle filter.

Chapter 5 seeks to address some of the issues discussed in this chapter. Classification is improved upon by including terrain information and more realistic prior probabilities for each target's speed. A different, less biased method is used to update the particle filter distribution using the classification from the cTBM.

Chapter 5

Using Terrain Information to Improve Joint Tracking and Classification

This chapter presents a new approach for JTAC with a WSN which we call 'tbmTerrain'. We have improved upon the work of Chapter 4 with the use of terrain information, more realistic prior probabilities, and a method for only assigning belief to targets that are feasible. Three scenarios, which are more realistic than the scenario of Chapter 4, are used for a performance evaluation; targets are modelled on real vehicles and have more complex, non-constant velocity trajectories. Roberts et al. (2010) is a condensed version of this chapter.

Novel improvements presented in this chapter are: the use of terrain information and how it restricts target movement; the use of an elliptical area to account for position uncertainty and the subsequent weighted combination of conditional plausibilities related to the terrain coverage within the ellipse; and the addition of a mechanism to prevent the assignment of belief to outcomes that past observations have shown are infeasible. Other improvements to further increase classification performance include the use of more realistic prior probabilities and the smoothing of the estimated target's speed.

An existing approach to JTAC with terrain information, by Powell and Marshall (2005), also uses of a particle filter and the TBM. Unfortunately Powell and Marshall do not consider the uncertainty of the target position when using terrain information — this can result in the incorrect assignment of belief when there is enough uncertainty to move the mean of the probability distribution from the true target position over a terrain boundary. However, Powell and Marshall perform multi-target tracking which we do not consider. Approaches that treat road networks as graphs (e.g. Ristic et al. (2004a, chap 10)) can allow for targets that travel both on-road and off-road; our approach goes further than this by considering how different types of off-road terrain affect the target movement by differing amounts (e.g. water vs. grass). A notable tracking-only approach

by Fosbury et al. (2007) uses 'trafficability' terrain information to deflect the target motion towards the direction which provides the least resistance to target motion.

Section 5.1 provides a brief overview of Intelligence Preparation of the Battlefield (IPB) — a methodology that has inspired our use of terrain information. Section 5.2 explains some of the new terminology and changes since Chapter 4. Section 5.3 explains the improvements and remaining terminology. The results are presented in Section 5.4. Finally, the conclusions and future work are discussed in Sections 5.5 and 5.6 respectively.

5.1 Background

Our method for incorporating terrain information into the JTAC is inspired by IPB. IPB (United States, 1994), also known as Intelligence Preparation of the Battlespace, is a methodology for analysing threats in a geographic area. As part of the IPB, terrain is analysed and labelled according to how it restricts unit movement. The intelligence used in terrain analysis includes, but is not limited to, information about vegetation, obstacles, terrain surface type, slope, and weather.

The IPB process divides terrain into three classes to indicate how restrictive movement will be — they are called 'Go', 'Slow Go', and 'No Go' (Talbot-Jones, P. Private communication). 'Go' terrain does not restrict unit movement — this could be a road for cars and may other types of wheeled or tracked vehicles. Terrain that is classed as 'Slow Go' provides some restriction to unit movement — for example, a person travelling over uneven or marshy ground. 'No Go' terrain is considered to be severely restricted — this could include lakes for non-amphibious vehicles such as cars.

5.2 Problem Formulation

This chapter builds upon Chapter 4, and as such much of the terminology and models are similar or the same. The same sensor selection algorithm, by Williams et al. (2007), is utilised. With the exception of resampling, the particle filter code remains unchanged. The sensor and communications costs models also remain unchanged.

The sensor selection algorithm, by Williams et al. (2007), is largely the same. It has been modified to only use sensors a minimum distance, $\min_d$, away from the expected position of target. This prevents the sensor measurements from being saturated when using a much higher SNR, which disrupts the target tracking.

The non-linear target trajectories used in this chapter are more realistic than that of Chapter 4 — targets slow down at turns and when travelling on more restrictive terrain.

Each target class, $\omega_i \in \Omega$, is based on a real unit; where possible, target characteristics (such as maximum speed) have been obtained from authoritative literature.

Terrain *types* such as road and grass are grouped into terrain *classes*; the set of terrain classes, $\mathcal{T} = \{\text{Go}, \text{SlowGo}, \text{NoGo}\}$ is the same as those used in IPB. However, our approach is not limited to three terrain classes, grouping terrain types into these classes provides a sufficient level of abstraction to simplify calculations yet still provide adequate richness for the classification process.

We create terrain maps based on the IPB approach to terrain analysis. A map, represented by a bitmap image, is segmented by terrain *type* (e.g. road, river, grass). Each segment of the same terrain type has the same colour. For each target class, a mapping is created from terrain type to terrain class — this allows each terrain type to affect target classes in different ways without requiring a separate map for each target class. Examples of terrain maps can be found in Section 5.4.

5.3 Improvements

The following improvements can by divided into two categories — those designed to improve classification performance, and those designed to improve feedback from classification to tracking.

5.3.1 Classification

The scenarios used in this chapter are more realistic than that of Chapter 4; for this reason, a more sophisticated method of classification is required. This is achieved by using a number of improvements to the method for calculating conditional belief masses (See Algorithm 5.1); these improvements consist of smoothing the estimated target speed, using terrain information, using a mechanism to ignore target classes are that no longer feasible, and using more realistic prior probabilities. This results in a more accurate classification.

5.3.1.1 Target speed

Classification takes into account both the target's speed, and the the terrain that it is travelling on. The target's speed is used in the same way to that of Chapter 4 with two notable exceptions — a different prior probability distribution is used, and the estimated speed of the target is smoothed using:

$$s_k = \frac{\mathbf{S}(\mu_k) + \mathbf{S}(\mu_{k-1}) + \ldots + \mathbf{S}(\mu_{k-\mathcal{W}+1})}{\mathcal{W}} \tag{5.1}$$

Algorithm 5.1 Calculating conditional masses

$[m_{1:k}(X|s_{1:k})]$ = **CalculateConditionalMasses** $[\mu_{k:k-w+1}, \mathbf{P}_k,$ terrain map$]$

$s_k = \frac{\mathbf{S}(\mu_k)+\mathbf{S}(\mu_{k-1})+\ldots+\mathbf{S}(\mu_{k-w+1})}{w}$ {See Equation 5.1}

Calculate the uncertainty ellipse for the current position using the mean and covariance of the a posteriori particle distribution and N_σ standard deviations

Overlay the uncertainty ellipse on the terrain map centred at $\mathbf{L}\mu_k$

for all $\omega_i \in \Omega$ **do**

  for all $\tau \in \mathcal{T}$ **do**

    Using the uncertainty ellipse, Calculate $f[\tau, \omega_i]$

    $pl_\tau(\omega_i|s_k) = (s_k - \bar{s_k})\, Bet f(s_k) + \int_{s_k}^{\infty} \left(1 - \frac{d\bar{a}}{da}\right) Bet f(a)\, da$ {See Equation 2.26}

  end for

  $pl_{\text{tbmTerrain}}(\omega_i|s_k) = \begin{cases} \sum_{\tau \in \mathcal{T}} f[\tau, \omega_i]\, pl_\tau(\omega_i|s_k) & \text{if } \omega_i \in \mathcal{F}_k; \\ 0 & \text{otherwise} \end{cases}$ {See Equation 5.4}

end for

for all $X \in 2^\Omega$ **do**

  $m_k(X|s_k) = \prod_{c_i \in X} pl_{\text{tbmTerrain}}(s_k|c_i) \prod_{c_i \in \bar{X}} [1 - pl_{\text{tbmTerrain}}(s_k|c_i)]$ {See Equation 2.25}

end for

$m_k(\Upsilon|s_k) = m_k(\Upsilon|s_k) + 1 - \sum_{X \in 2^\Omega} m_k(X|s_k)$

for all $X \in 2^\Omega$ **do** {Calculate fused masses using PCR}

  $m_{1:k}(X|s_{1:k}) = m_{1:k-1 \cap k}(X) + \sum_{\substack{Y \in 2^\Omega \setminus \{X\} \\ X \cap Y = \emptyset}} \left[\frac{m_{1:k-1}(X)^2 m_k(Y)}{m_{1:k-1}(X) + m_k(Y)} + \frac{m_k(X)^2 m_{1:k-1}(Y)}{m_k(X) + m_{1:k-1}(Y)} \right]$ {See Equation 2.33}

end for

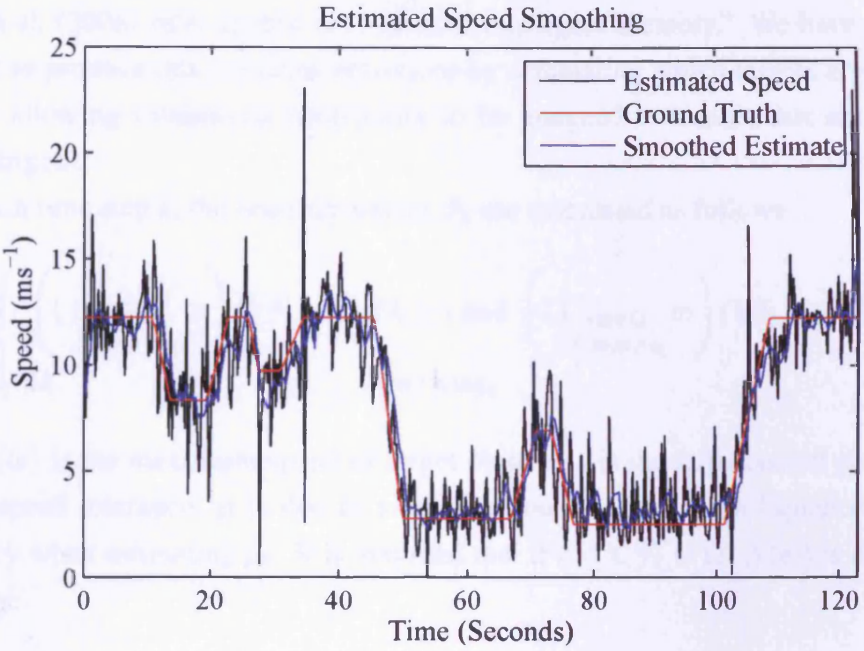


Figure 5.1: The real, estimated, and smoothed (using Equation 5.1) target speed from a typical Monte Carl trial for Scenario A ($T = 0.25$ and $\mathcal{W} = 7$).

where μ is calculated using Equation 4.9, and $\mathcal{W}$ is the window size. A smoothed estimate is required because the estimated speed for a single time step is typically too noisy for accurate classification — especially when tracking a target with non-linear dynamics (See Figure 5.1). Smoothing using Equation 5.1 will result in some bias, but with a sufficiently small value of T and $\mathcal{W}$ it does not negatively impact classification.

5.3.1.2 Utilising Terrain Information

Terrain information is used in the classification process to take into account how the underlying terrain may hinder a target's progress. An uncertainty ellipse at N_σ standard deviations is used to calculate the underlying terrain — this provides a more robust classification than using a single point at $\mathbf{L}\mu_k$ as the terrain may vary over a small area and there will usually be some inaccuracy as to the exact location of the target. The proportions of each terrain class within the ellipse are stored in f, a $|\mathcal{T}|$ by $|\Omega|$ array, such that $\sum_{\tau \in \mathcal{T}} f[\tau, \omega_i] = 1 \; \forall \; \omega_i \in \Omega$. The notation $f[\tau, \omega_i]$ is used to denote the proportion of the ellipse coverage that is of the terrain class τ for the target class ω_i.

5.3.1.3 Ignoring Infeasible Target Classes

One problem that exists with using the TBM to recursively fuse *bbms* is that belief can be assigned to a target class that from earlier behaviour is obviously not feasible —

Powell et al. (2006) refer to this as a "lack of intelligent memory." We have modified the TBM to produce this expected behaviour by calculating which targets are feasible, and only allowing conditional plausibility to be assigned to masses that are only for feasible targets.

At each time step k, the feasible classes $\mathcal{F}_k$ are calculated as follows:

$$\mathcal{F}_k = \begin{cases} \left(\bigcup_{\substack{\forall \omega \in \Omega, \\ \mathcal{S}(\omega)\rho \geq s_k}} \omega \right) \cap \mathcal{F}_{k-1} & \text{if } k \geq \iota \text{ and } \left(\bigcup_{\substack{\forall \omega \in \Omega, \\ \mathcal{S}(\omega)\rho \geq s_k}} \omega \right) \cap \mathcal{F}_{k-1} \neq \emptyset; \\ \Omega & \text{otherwise,} \end{cases} \tag{5.2}$$

where $\mathcal{S}(\omega)$ is the maximum speed of target class ω, ι is the initialisation period, and ρ is the speed tolerance; ρ is due to a small amount of bias from Equation 5.1 and inaccuracy when estimating μ_k. It is assumed that if $k \leq \iota$, $\mathcal{F}_k = \Omega$. $\mathcal{S}(\omega)$ is calculated as follows:

$$\mathcal{S}(\omega) = \begin{cases} \mathcal{S}_{\text{Go}}(\omega) & \text{if } f[\text{Go}, \omega] > 0; \\ \mathcal{S}_{\text{SlowGo}}(\omega) & \text{if } f[\text{Go}, \omega] = 0 \text{ and } f[\text{SlowGo}, \omega] > 0; \\ \mathcal{S}_{\text{NoGo}}(\omega) & \text{otherwise.} \end{cases} \tag{5.3}$$

$\mathcal{S}_{\text{Go}}(\omega)$, $\mathcal{S}_{\text{SlowGo}}(\omega)$, and $\mathcal{S}_{\text{NoGo}}(\omega)$ are the maximum speeds for target class ω whilst travelling over terrain classes Go, Slow Go, and No Go respectively. $\mathcal{F}_k$ is then used when calculating conditional plausibility to ignore target classes that are infeasible (See Equation 5.4).

5.3.1.4 More Realistic Prior Probabilities — speedPDF

In Chapter 4, a Gaussian pignistic density was used, resulting in conditional plausibility, conditional on target speed, which was calculated using Equation 4.18; this was sufficient for a constant velocity target trajectory but it is not suitable for more realistic target trajectories. The pignistic density function, $Betf$, used in this chapter is asymmetric; there is a slow increase in $Betf(y)$ from $y = 0$ to the expected maximum speed of the target, and then a sharp decline from the expected maximum speed toward ∞. We call this new pignistic density function a 'speedPDF'; examples of this can found in Figures 5.3b–5.3d later in this chapter.

A speedPDF is based on a Gamma density. The shape parameter is fixed, and the density is transformed to make the long tail pass through the y-axis. Ideally at $y = 0$, $Betf(y) = 0$ but this is not possible. It is accepted that when $y = 0$, $Betf(y) \simeq accp$ where $accp$ is small value. The maximum estimated speed, $\mathcal{S}_\tau(\omega)$, is then used to fix the mode of the Gamma density for the target class ω and terrain class τ. An iterative

method is used to calculate the scale parameter with the given shape and mode constraints. An analytical solution is not possible as there is no known analytical solution to the inverse Gamma function. The conditional plausibility is then calculated from $Bet f$ using Equation 2.26.

Basing the speedPDF on a Gamma density is not a perfect solution, but it is better than the alternatives. A Gaussian distribution does not adequately reflect the nature of the target dynamics, and has similar problems to the Gamma density in that when $y = 0, Bet f(y) \neq 0$. A triangular distribution would provide a simpler solution, but it's 'sharp' peak is inferior to the 'plateau effect' that occurs with the Gamma density. This effect is useful because typically a target spends a considerable amount of time near this speed; the peak of the triangular distribution would result in too large a change in $Bet f(y)$ for small changes in y, possibly resulting in the assignment of belief in an erratic manner. Other alternative approaches could have been to use α-stable distributions (Fiche et al., 2010) or a mixture of Gaussians (Caron et al., 2006) for our pignistic density function.

5.3.1.5 Combining Plausibilities

In order to create *bbms* from the estimated target speed and the underlying terrain, it is necessary to combine conditional plausibilities. Conditional plausibility is calculated for each terrain and target class given the estimated target speed; it is then combined for each target class resulting in a conditional plausibility for each target class. Using the combined conditional plausibility results in a *bba* that takes into account not only target speed and the terrain the target is travelling over, but also the uncertainty of the target's kinematic state.

The conditional plausibilities are combined as follows:

$$pl_{\text{tbmTerrain}}(\omega|s_k) = \begin{cases} \sum_{\tau \in \mathcal{T}} f[\tau, \omega]\, pl_\tau(\omega|s_k) & \text{if } \omega \in \mathcal{F}_k; \\ 0 & \text{otherwise,} \end{cases} \tag{5.4}$$

where $pl_\tau(\omega|s_k)$ is the conditional plausibility calculated for target ω using priors for the terrain class τ, and speed s_k. $pl_{\text{tbmTerrain}}$ can then be used to create conditional *bbms* for time k using Equation 2.25.

Any remaining mass that would usually be assigned to the empty set is added to the most uncertain (yet still feasible) set, Υ — this is required as PCR, which is used to fuse belief masses, does not allow an input *bba* with $m(\emptyset) > 0$. The mass is assigned to the most uncertain set as it is a better candidate than Ω if some target classes are infeasible. This is also a better method than normalising the *bba* as it may give a false indication

of the true outcome. Once the *bba* has been calculated for time k, it is fused with the existing *bba* which was created using the target state estimate for times 1 to $k-1$. The resulting *bba* is used to calculate *BetP*.

5.3.2 Transferable Belief Model Feedback to the Particle Filter

In addition to improving the classification process, feedback from classification to the particle filter has been improved. A new method for this feedback is presented here and outlined in Algorithm 5.2. The aim of this new method is to use information from the classification process to reduce the uncertainty of the target position.

Algorithm 5.2 Feedback from classification to the particle filter

$\left[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_p}\right] =$ **Feedback** $\left[BetP, \mu_k, \mathbf{P}_k, \{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_p}, f\right]$
  $\omega_i = \arg\max_{\omega_i \in \Omega} BetP(\omega_i)$
  if $N_{\text{eff}} < N_{\text{effThres}}$ **then** {Is resampling required?}
    if not $((f[\text{Go}, \omega_i] = 1)$ **or** $(f[\text{SlowGo}, \omega_i] = 1)$ **or** $(f[\text{NoGo}, \omega_i] = 1))$ **and** $k \leq \iota$ **then**

$$\bar{\mathbf{P}}_k = \begin{bmatrix} \kappa & \kappa & \kappa & \kappa \\ \kappa & 1 & \kappa & 1 \\ \kappa & \kappa & \kappa & \kappa \\ \kappa & 1 & \kappa & 1 \end{bmatrix} .* \mathbf{P}_k \text{ \{See Equation 5.5\}}$$

    for $i = 1$ **to** N_p **do** {Perform parametric resampling (See Equation 5.6)}
      $\mathbf{x}_k^i \sim \mathcal{N}(\mathbf{x}_k^i; \mu_k, \bar{\mathbf{P}}_k)$
      $w_k^i = 1/N_p$
    end for
  else
    $\left[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_p}\right] =$ **SystematicResample** $\left[\{\mathbf{x}_k^i, w_k^i\}_{i=1}^{N_p}\right]$ {See Algorithm 2.1}
  end if
end if

When required, feedback to the particle filter is achieved by updating the covariance matrix, $\mathbf{P}_k$, and then sampling a new set of particles from a multivariate Gaussian distribution parameterised by μ_k and the updated covariance matrix. If there is more than one terrain class covered by the uncertainty ellipse then the parts of the covariance matrix related to position are reduced, resulting in a conditioned covariance matrix, $\bar{\mathbf{P}}_k$. Assuming that the state vector is $[pos_x\ vel_x\ pos_y\ vel_y]^T$, $\bar{\mathbf{P}}_k$ is calculated as follows:

$$\bar{\mathbf{P}}_k = \begin{bmatrix} \kappa & \kappa & \kappa & \kappa \\ \kappa & 1 & \kappa & 1 \\ \kappa & \kappa & \kappa & \kappa \\ \kappa & 1 & \kappa & 1 \end{bmatrix} .* \mathbf{P}_k, \tag{5.5}$$

where κ is the conditioning factor, and $.*$ is the notation used to denote element-by-element multiplication (the same notation used by Matlab). After calculating $\bar{\mathbf{P}}_k$, resampling is performed. Each particle, $\mathbf{x}_k^i$, and it's respective weight, w_k^i, is sampled and calculated respectively, as follows:

$$\mathbf{x}_k^i \sim \mathcal{N}\left(\mathbf{x}_k^i; \mu_k, \bar{\mathbf{P}}_k\right), \; w_k^i = 1/N_p, \tag{5.6}$$

where μ_k is calculated using Equation 4.9. We call the above resampling method 'parametric resampling'.

Parametric resampling does not preserve the potentially complex and non-Gaussian nature of the particle distribution — this is especially the case in the scenarios used in this thesis where only range sensors are utilised. This makes it inferior to systematic resampling that is used in Chapter 4, but it does provide a way to 'reshape' the distribution in a straightforward manner. A hybrid approach is used that results in a compromise between the two resampling methods: parametric resampling is used when conditioning is required, systematic resampling is used in all other cases.

As the new resampling method has the potential to remove some of the detail from the particle distribution, the particle filter has been modified to resample less often. Resampling is performed at each time step only if the number of effective particles is lower than a threshold, $\bar{N}_{eff} < N_{thr}$, where $\bar{N}_{eff}$ is the estimated effective sample size (See Equation 2.1.2.2). This results in conditioning taking place less often, but it aims strikes a balance between providing feedback from classification to tracking, and preserving the complexity of the particle distribution.

5.4 Results

Three scenarios (A, B, and C) were used to test the improvements presented in this chapter. For each scenario our new approach, tbmTerrain, was compared with our previous approach (See Chapter 4) and the approach of Williams et al. (2007). As with Chapter 4, horizon lengths of 5, 10, and 25 are used, and 100 Monte Carlo trials were used for each combination of parameters.

A different sensor layout was used for each of the 100 Monte Carlo trials in each scenario, and the same set of 100 sensor layouts was used for each combination of parameters. A single target trajectory was used for each scenario as opposed to a different target trajectory for each trial (as per Chapter 4) — this was because random variations in the target's kinematic state could have resulted in a trajectory that travels too fast or over the wrong terrain type. The potential pitfalls of using a single target trajectory

for each scenario have been mitigated by using 100 different sensor layouts — these were generated by using the same strategy utilised in Chapter 4 but for a different sized region.

The terrains created were based on real geographic areas using Ordnance Survey map data (Ordnance Survey, 2005, 2007). A different target class was the true class for each scenario. Each scenario provides a different view on the performance and characteristics of the improvements presented in this chapter. In Scenario A (Section 5.4.1), an amphibious light tank drives along a road, then over a river and grass. In Scenario B (Section 5.4.2), a car drives up to a roundabout, around it, then away from it — remaining on the road surface at all times. In Scenario C (Section 5.4.3), a main battle tank initially travels along a road, and then off-road. The assumptions made about the potential speeds of each target class can be found in Table 5.1 and the effect of the each terrain type on the performance of each target class can be found in Table 5.2. Figures 5.2 and 5.3 show the conditional pignistic density function that is calculated for each target class and terrain class, the conditional pignistic density for the approach presented in Chapter 4 is also shown.

The target state and measurement model from Chapter 4 was used in all of the scenarios in this chapter; some of the parameters have been changed to more suitable values. The intensity of the process noise of the target state, q, was increased to 50 as a white noise acceleration model was used to allow for the target maneuverability. The signal to noise ratio has been improved (by increasing a, b was not changed) to allow 20 or 30 sensors (depending on the scenario) to cover a much larger region. It was not possible to use more sensors instead of increasing a due to the computational complexity of the sensor selection algorithm. The minimum sensor selection distance was $\min_d = 50$ m. q, a, and $\min_d$ were all determined experimentally. The Euclidean distance measure was used for this.

The particle filter has been modified to use a larger spread of particles on initialisation. In all scenarios, a resampling threshold, N_{thr}, equivalent to 66% (similar to Doucet et al. (2001b, p. 333)) and, as with Chapter 4, a sampling interval of $T = 0.25$ was used.

For all of the simulations that used our previous approach, a condition factor of $\gamma = 3.5$ was used. Instead of creating a *bba* for each particle, creating N_p *bbas* for each time step, we have created a single *bba* at each time step using μ_k — this was changed to reduce computational requirements.

Target Class		Maximum Speed (ms^{-1})		
Type	Name	Go	Slow Go	No Go
Pedestrian	Pedestrian	1.3411	0.4470	0.1118
Amphibious Light Tank	PT76	12.2222	2.5000	0.4470
Light Tank	T62	13.8889	3.5763	0.4470
Main Battle Tank	Challenger 1	15.5556	4.4704	0.4470
Car	ZiL 41041	20.1168	6.7056	0.4470
Bicycle	Bicycle	4.4704	1.3411	0.2235

Table 5.1: Target classes and their assumed maximum speeds for each terrain class. The bicycle class is only used in a modified version of Scenario A (See page 78).

Target Class	Road	Grass	Water	Buildings	Marsh	Trees	Steep Land
Pedestrian	G	G	N	N	S	G	S
PT76	G	G	S	N	S	S	N
T62	G	G	N	N	N	S	N
Challenger 1	G	G	N	N	N	S	N
ZiL 41041	G	S	N	N	N	S	N
Bicycle	G	S	N	N	N	S	N

Table 5.2: The assumed best performance of each target class for each terrain type. The letters 'G', 'S', and 'N' indicate Go, Slow Go, and No Go respectively. For example, a pedestrian travelling on the terrain types 'Road' and 'Grass' is considered unhindered and can therefore travel up to and including it's 'Go' speed — it's maximum possible speed.

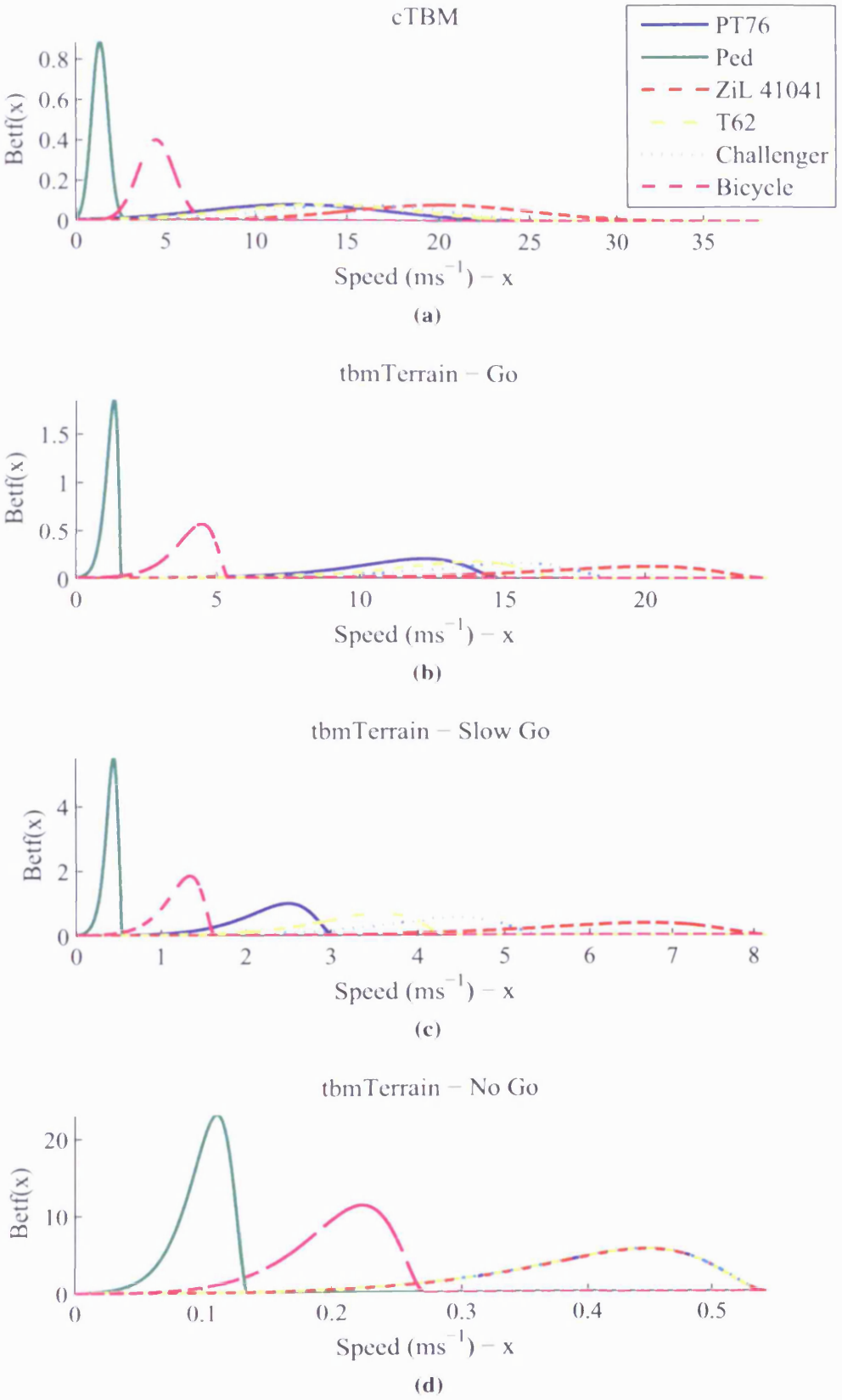


Figure 5.3: *Betf* for the cTBM (a) and tbmTerrain terrain classes 'Go' (b), 'Slow Go' (c), and 'No Go' (d). Different scales have been used in order to show as much detail as possible.

5.4.1 Scenario A

Scenario A consists of an amphibious light tank moving from the top right of a 630m by 436m region to the bottom of the map (See Figure 5.4). The target travels over road, grass, and water. 20 sensors are used in each Monte Carlo trial, with $R = 1.75$ and $a = 1500000$. Each simulation is 493 time steps long. The tbmTerrain trials used an initialisation period of $\iota = 30$, a speed smoothing window size of $\mathcal{W} = 7$, and a speed tolerance of $\rho = 1.75$. These parameters were determined experimentally.

Figure 5.5 provides a summary performance comparison. A comparison of classification performance can be found in Figure 5.6, tracking performance in Figure 5.7, and communications cost accrual in Figure 5.8. As with Chapter 4, the box plots denote the median with a red vertical line, the box contains the interquartile range, the whiskers extend to the largest and smallest non-outlier data points, and a '+' is used to denote outliers. Outliers are less than $q_1 - 1.5(q_3 - q_1)$ or more than $q_3 + 1.5(q_3 - q_1)$, where q_1 and q_3 are the first and third quartiles, respectively.

From the results shown in Figure 5.6 it appears that our previous approach to classification is superior to our new approach — but this is not the case; with our previous approach and the priors in Figure 5.3a, any *bba* created for a target travelling between approximately 2.6 and 13 ms^{-1} will be result in a *BetP* with the most likely outcome being an amphibious light tank (See Figure 5.13). The target travels between these speeds for approximately 80% of this scenario (See Figure 5.1).

This behaviour can be confirmed by comparing the partial confusion matrices of the two JTAC approaches (See Tables 5.3 and 5.4). The tbmTerrain approach classifies the target as the correct class, as a light tank, and as a car. The JTAC of Chapter 4 only classifies the target as a car once, but it also incorrectly classifies the target as a pedestrian, a light tank, and a main battle tank . Both apporaches classify the target as a light tank when the speed of the target is overestimated. The partial confusion matrices in this chapter have been created by adding up the number of time steps that each class has the highest probability. This is performed for all simulations using the same JTAC approach. Full confusion matrices are not shown as it would require the simulations to be run for every target class — requiring substaintially more simulations.

The results of a modified version of this scenario, which includes an extra target class, can be found on page 78; the use of an extra target class reduces the superior, yet coincidental performance of our previous approach whilst having little or no impact on the performance of the approach of this chapter.

The tracking performance is similar across all approaches for the same horizon length; this may be due to a combination of our new approach using an inferior resampling method when feedback from classification to tracking occurs, and using a

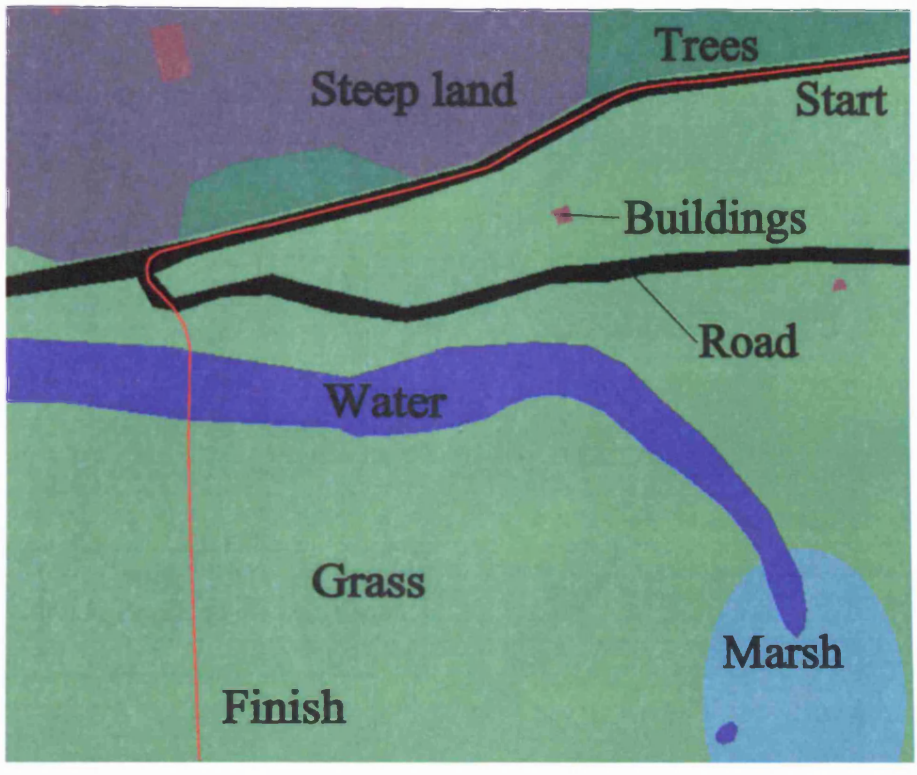


Figure 5.4: Target trajectory and terrain for Scenario A. Annotations have been added to the terrain map. The red line is the target trajectory.

fairly simplistic method for providing the feedback. As tracking performance is similar for all approaches with the same horizon length, the sensor usage is similar too (See Figure 5.9), resulting in similar communications costs.

An overview of the typical behaviour of the TBM with our new approach for this scenario can be found in Figures **??**–5.12. Figure **??** shows the *bba* created from the target state estimate at each time step, i.e. the non-fused masses. Figure 5.11 shows the fused belief masses for each time from using Algorithm 5.1. Figure 5.12 shows the classification output from the TBM for each step. For this scenario, it can be seen that from the start of the simulation up to approximately time step 280, each non-fused *bba* has a large amount of uncertainty which is reduced after successive combinations with PCR. Then for a short period of time the target is incorrectly classified as a car — this is because the target is travelling over grass at a speed that gives a significant amount of plausibility to the car. As soon as the uncertainty ellipse only covers water, $\mathcal{F}_k$ only contains the amphibious light tank — this results in the sharp change in *bbas* created from approximately time step 310 onwards.

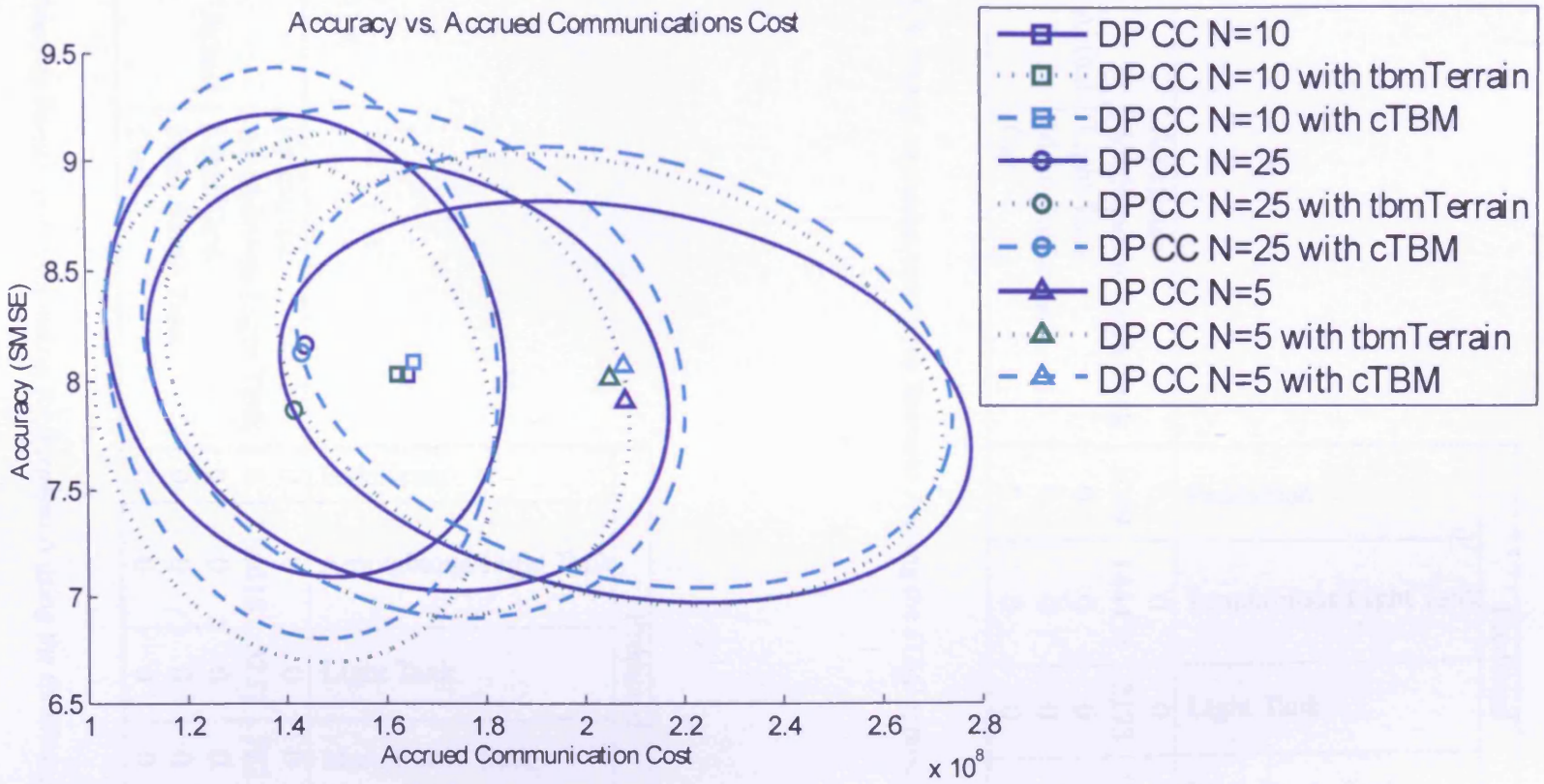


Figure 5.5: An overview of track accuracy and accrued communications costs for Scenario A.

		Predicted				
		Pedestrian	Amphibious Light Tank	Light Tank	Main Battle Tank	Car
Actual	Pedestrian	0	0	0	0	0
	Amphibious Light Tank	1069	144471	2173	186	1
	Light Tank	0	0	0	0	0
	Main Battle Tank	0	0	0	0	0
	Car	0	0	0	0	0

Table 5.3: Partial confusion matrix for Scenario A using the JTAC approach of Chapter 4.

		Predicted				
		Pedestrian	Amphibious Light Tank	Light Tank	Main Battle Tank	Car
Actual	Pedestrian	0	0	0	0	0
	Amphibious Light Tank	0	134618	5237	722	7323
	Light Tank	0	0	0	0	0
	Main Battle Tank	0	0	0	0	0
	Car	0	0	0	0	0

Table 5.4: Partial confusion matrix for Scenario A using the tbmTerrain approach.

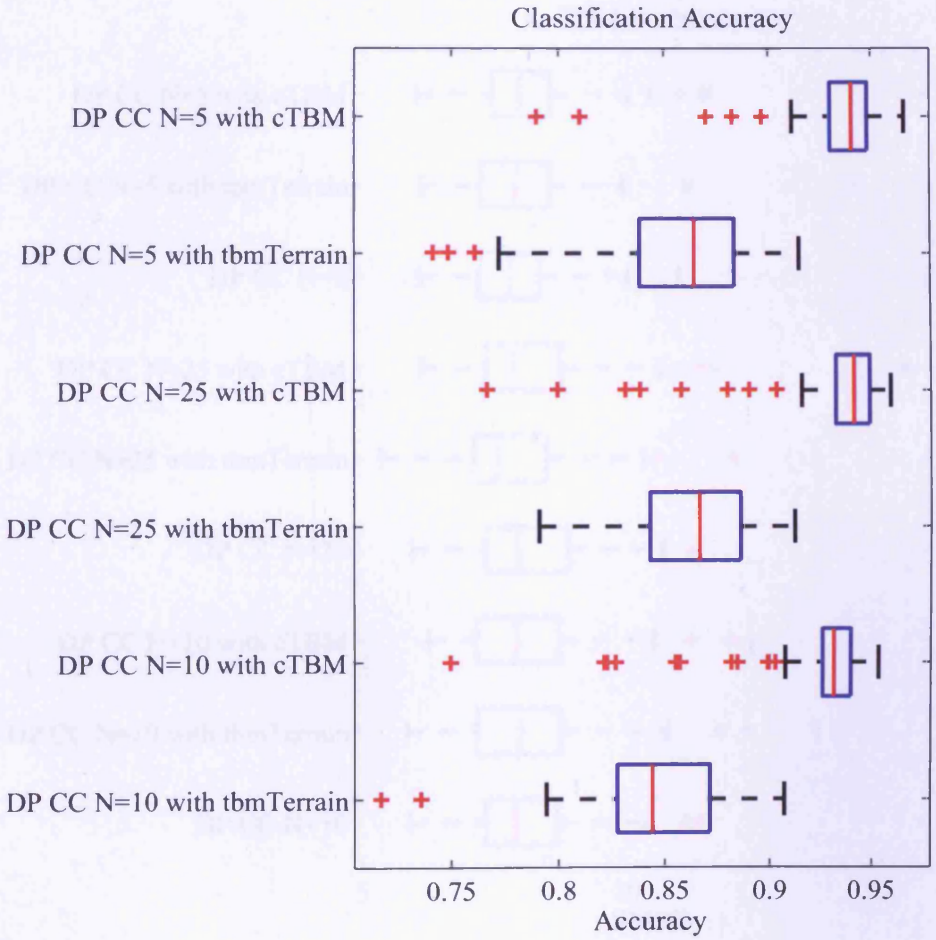


Figure 5.6: Classification accuracies for Scenario A. Each box plot is created from the mean value of $BetP(A)$ over time for each simulation.

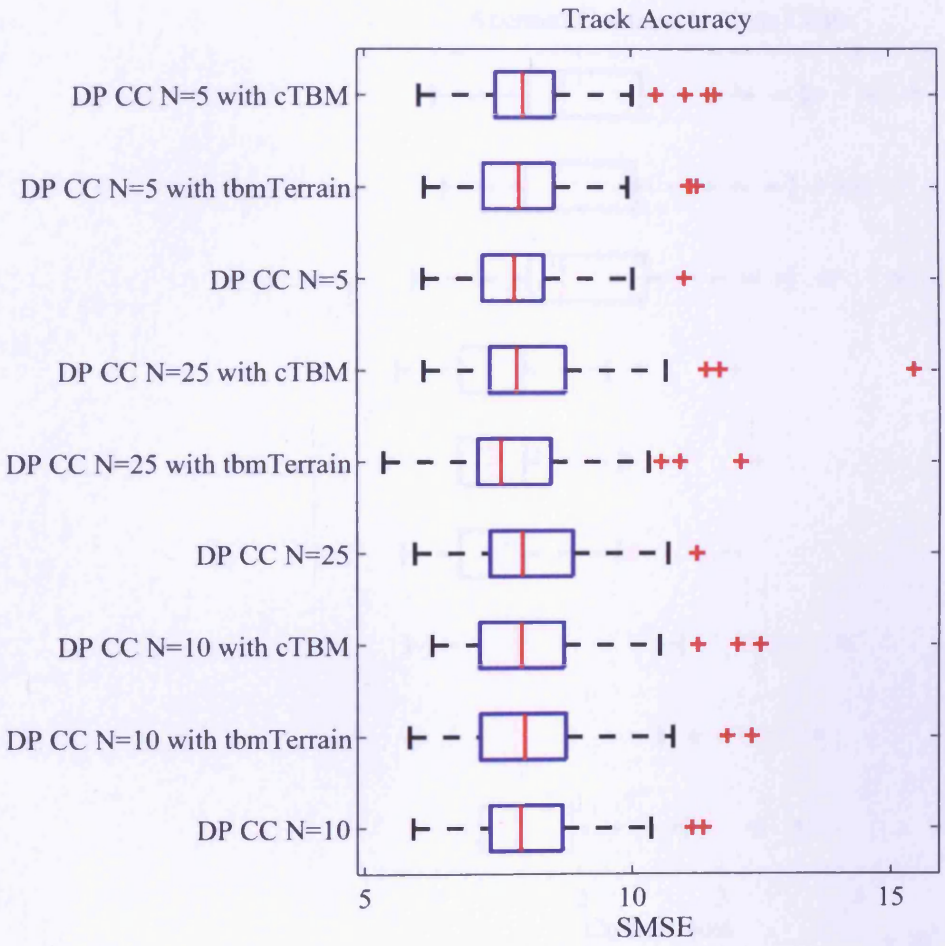


Figure 5.7: A comparison of track accuracies for Scenario A. Each box plot is created from a set of 100 Monte Carlo trials.

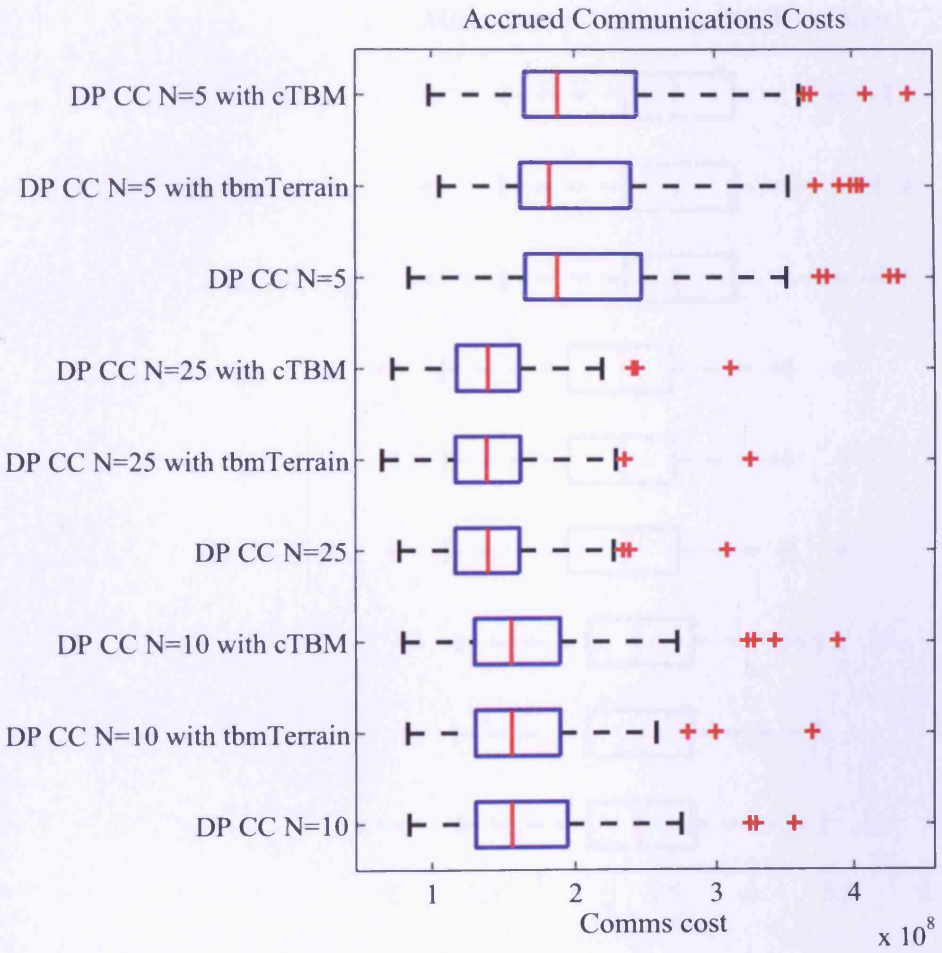


Figure 5.8: Accrued communications costs for Scenario A. Each box plot is created from a set of 100 Monte Carlo trials.

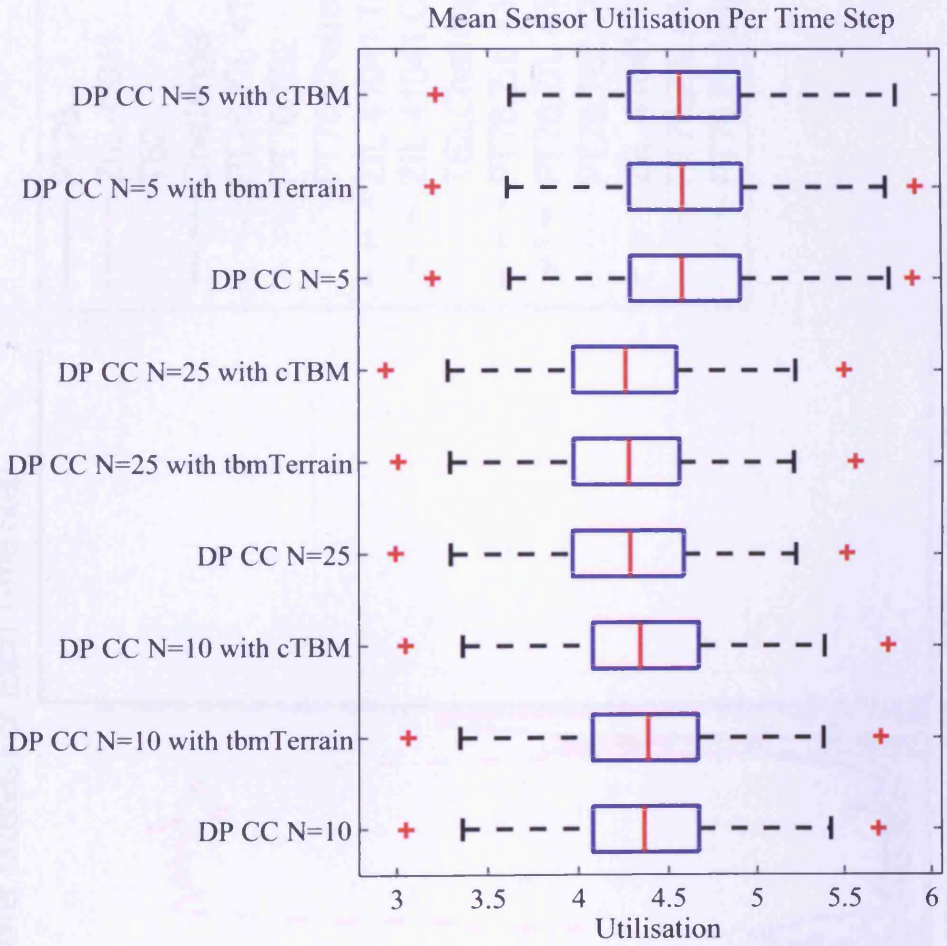


Figure 5.9: Mean number of sensors used per time step for Scenario A. Each box plot is created from a set of 100 Monte Carlo trials.

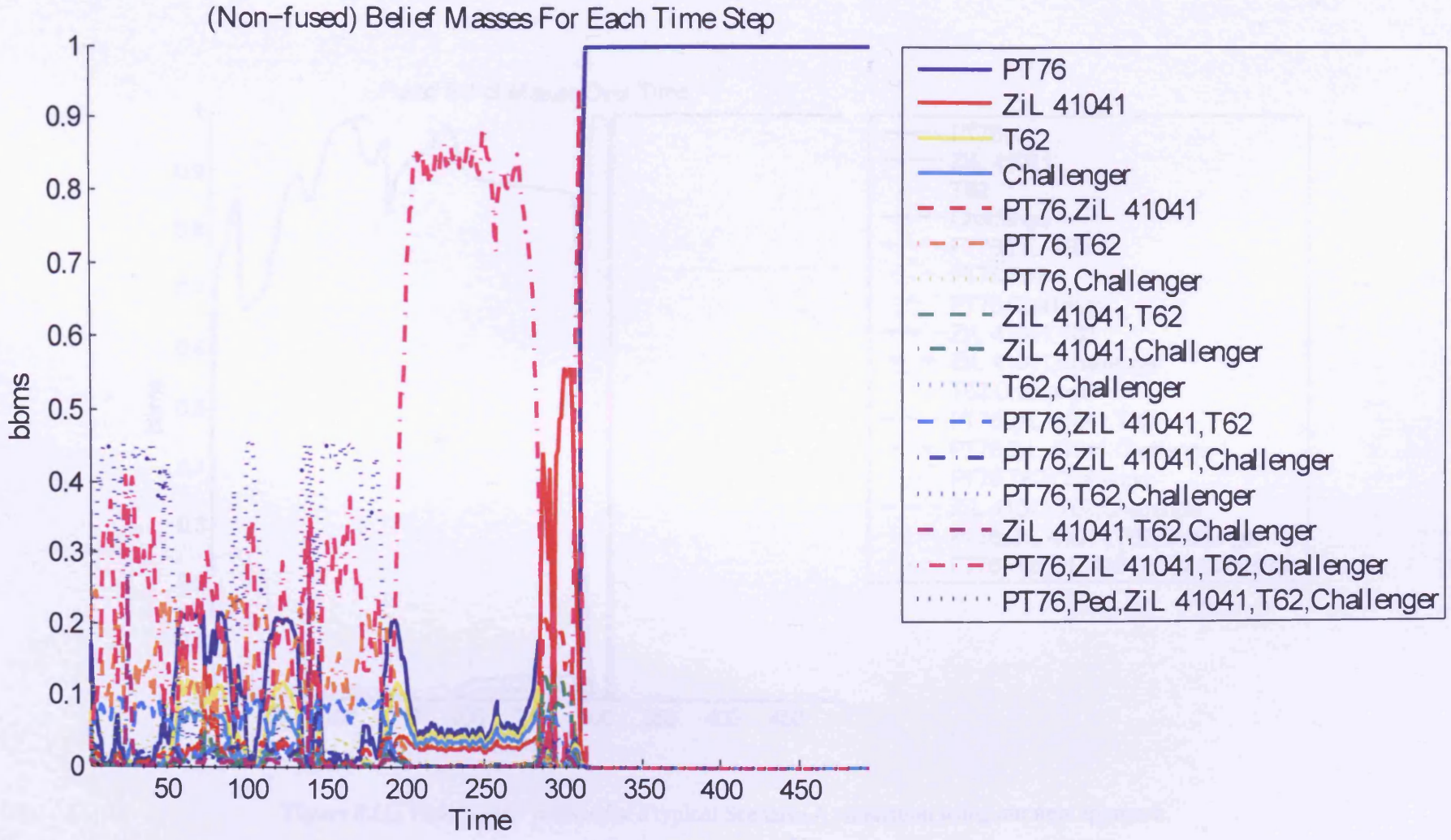


Figure 5.10: Non-fused belief masses for a typical modified Scenario A simulation using our new approach.

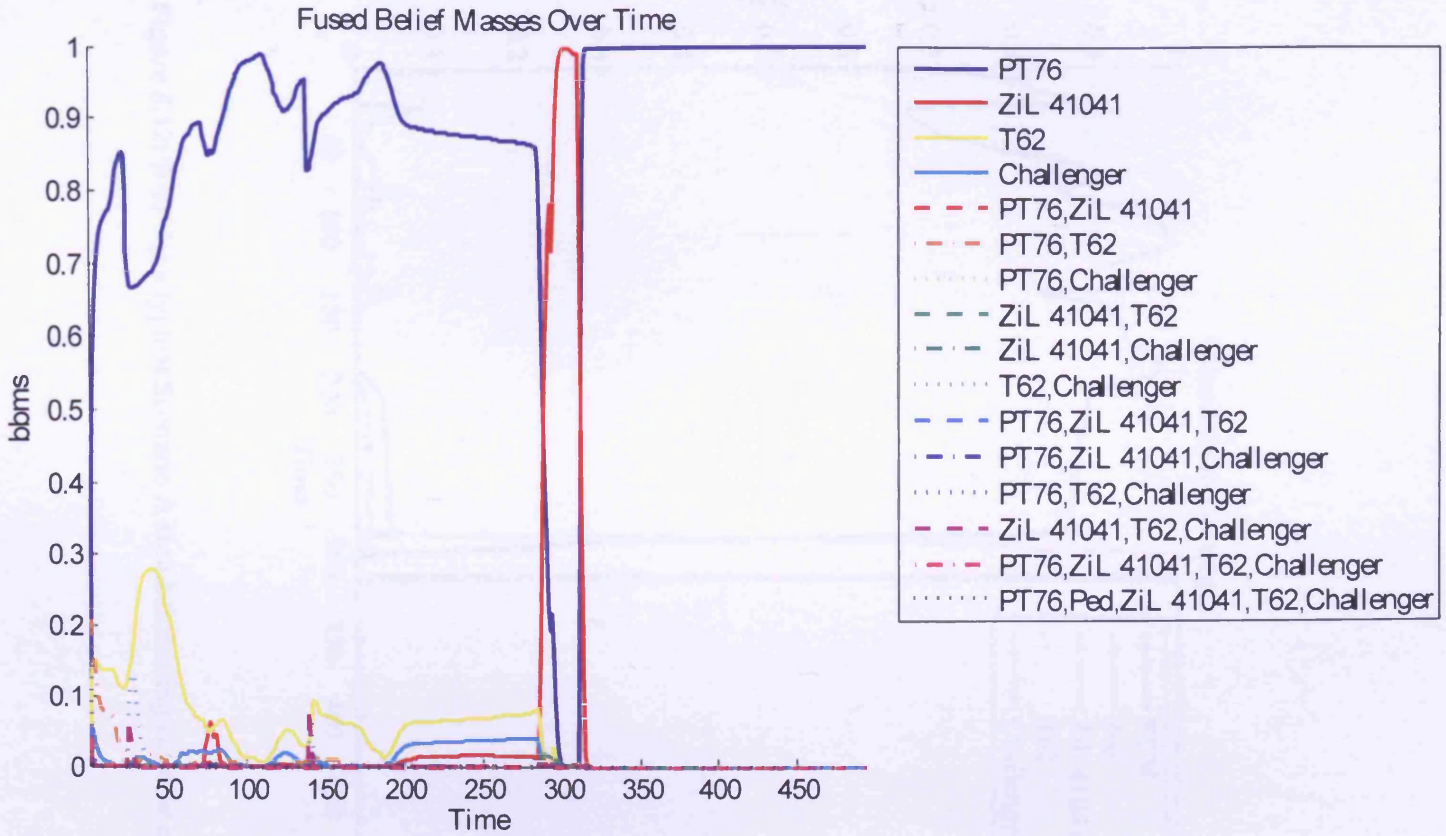


Figure 5.11: Fused belief masses for a typical Scenario A simulation using our new approach.

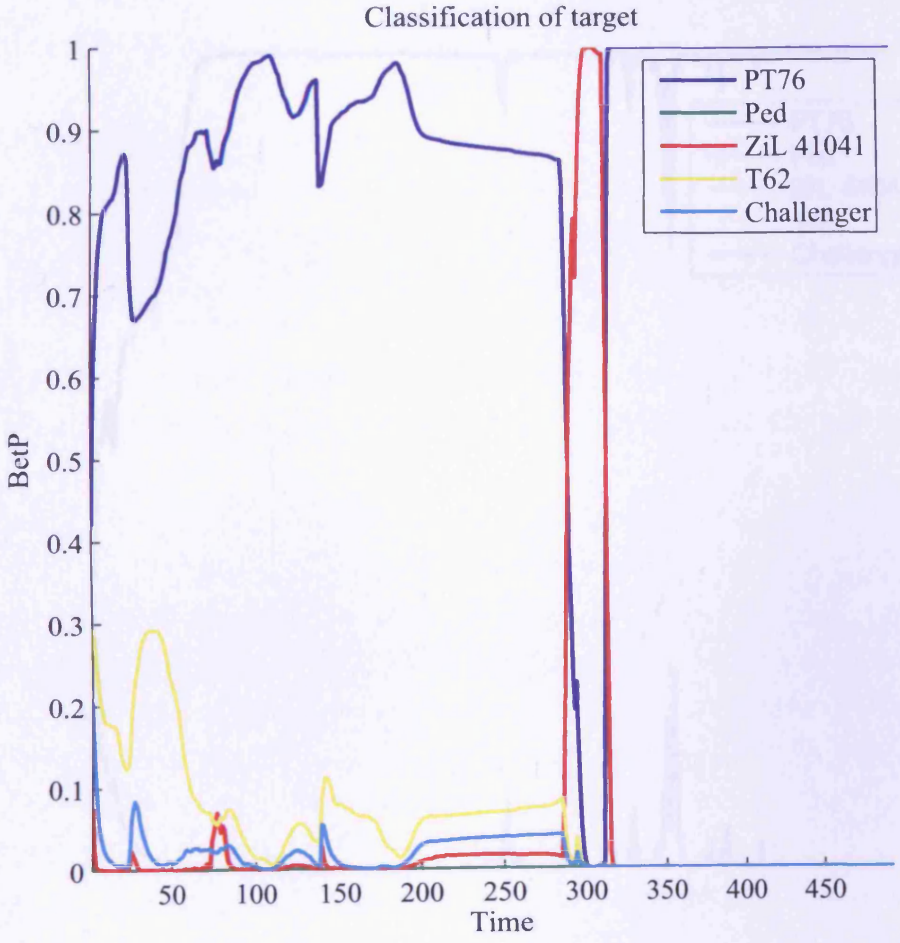


Figure 5.12: *BetP* for a typical Scenario A simulation using our new approach.

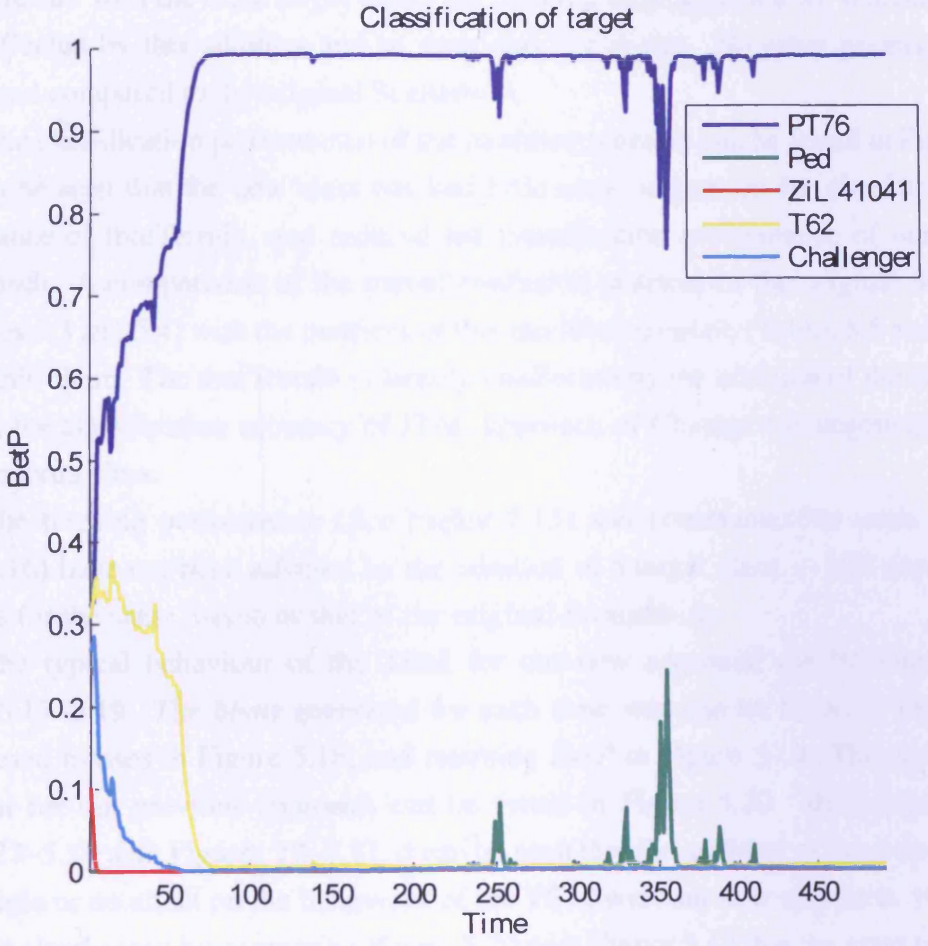


Figure 5.13: *BetP* for a typical Scenario A simulation using the approach of Chapter 4.

5.4.1.1 Modified Scenario

A modified version of Scenario A was created to demonstrate the coincidental nature of the superior performance of our previous approach compared to the approach of this chapter. A bicycle was been added to Ω; this was to reduce effect found in Scenario A where the classification performance of our previous approach appears to be better than that of tbmTerrain because of the choice classes in Ω and their respective conditional prior probabilities. The simulations for both tbmTerrain trials and our previous approach were re-run with the extra target class; the tracking only approach by Williams et al. is not affected by this addition and as such was not re-run. No other parameters were changed compared to the original Scenario A.

The classification performance of the modified scenario can be found in Figure 5.14; it can be seen that the new class has had little or no impact on the classification performance of tbmTerrain, and reduced the classification performance of our previous approach. A comparision of the partial confusion matrices of the original Scenario A (Tables 5.3 and 5.4) with the matrices of this modified scenario (Tables 5.5 and 5.6) support this claim. The tbmTerrain is largely unaffected by the addition of the extra target class; the classification accuracy of JTAC approach of Chapter 4 is negatively affected by the extra class.

The tracking performance (See Figure 5.15) and communication costs (See Figure 5.16) have not been affected by the addition of a target class — it is expected that this is for the same reason as that of the original Scenario A.

The typical behaviour of the TBM for our new approach can be found in Figures 5.17–5.19. The *bbms* generated for each time step can be found in Figure 5.17, the fused masses in Figure 5.18, and resulting *BetP* in Figure 5.19. The classification output for our previous approach can be found in Figure 5.20. By comparing Figures ??–5.19 with Figures ??–5.12, it can be seen that the addition of the extra class has had little or no effect on the behaviour of the TBM with our new approach. However it can be clearly seen by comparing Figure 5.20 with Figure 5.13 that the extra target class has a substantial effect on the behaviour of the TBM using the approach of Chapter 4. Figure 5.20 shows two periods of confusion — between approximately time steps 200–280 and 300-425, the target slows down (To perform a sharp turn and to cross water respectively) enough to assign more belief to the target class being a bicycle instead of an amphibious tank. This affect is not seen with tbmTerrain due to the combined usage of better prior probabilities and only assigning belief to feasible target classes. It is the behaviour seen in Figure 5.20 that has resulted in a reduced classification performance for approach of Chapter 4.

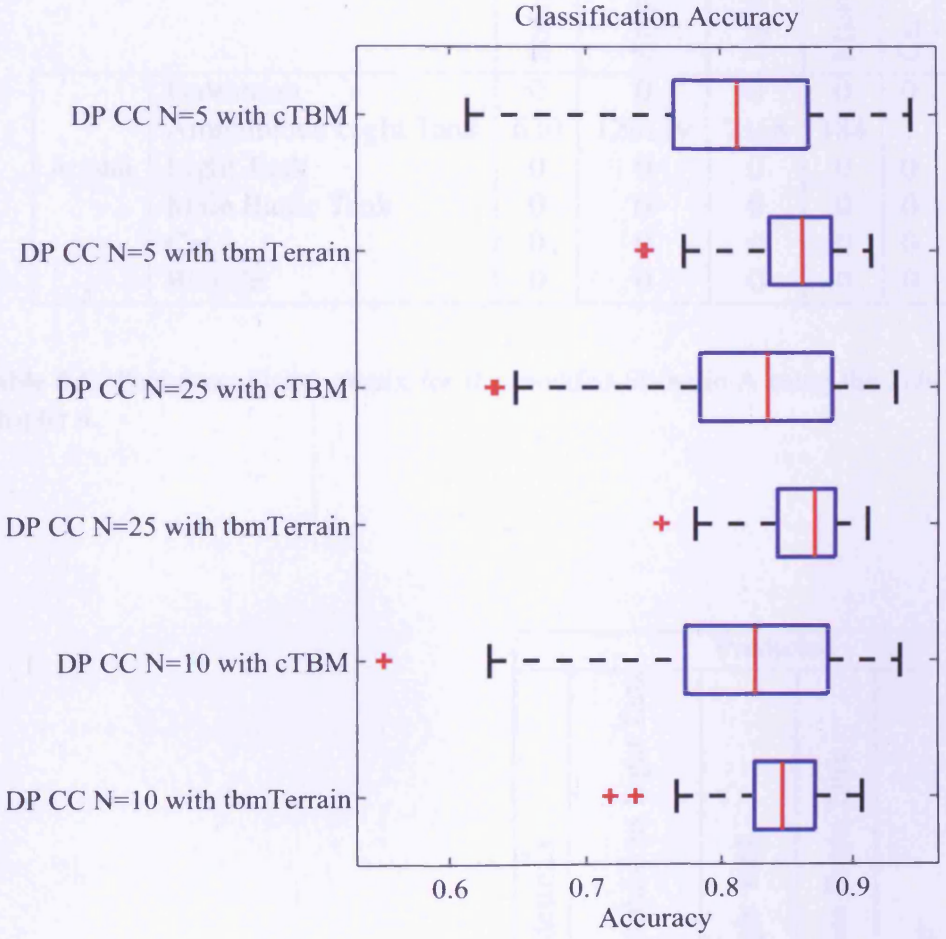


Figure 5.14: Classification accuracies for the modified Scenario A. Each box plot is created from the mean value of $BetP(A)$ over time for each simulation.

		Predicted					
		Pedestrian	Amphibious Light Tank	Light Tank	Main Battle Tank	Car	Bicycle
Actual	Pedestrian	0	0	0	0	0	0
	Amphibious Light Tank	630	128119	2168	184	1	16798
	Light Tank	0	0	0	0	0	0
	Main Battle Tank	0	0	0	0	0	0
	Car	0	0	0	0	0	0
	Bicycle	0	0	0	0	0	0

Table 5.5: Partial confusion matrix for the modifed Scenario A using the JTAC approach of Chapter 4.

		Predicted					
		Pedestrian	Amphibious Light Tank	Light Tank	Main Battle Tank	Car	Bicycle
Actual	Pedestrian	0	0	0	0	0	0
	Amphibious Light Tank	0	134355	5305	751	7482	7
	Light Tank	0	0	0	0	0	0
	Main Battle Tank	0	0	0	0	0	0
	Car	0	0	0	0	0	0
	Bicycle	0	0	0	0	0	0

Table 5.6: Partial confusion matrix for the modified Scenario A using the tbmTerrain approach.

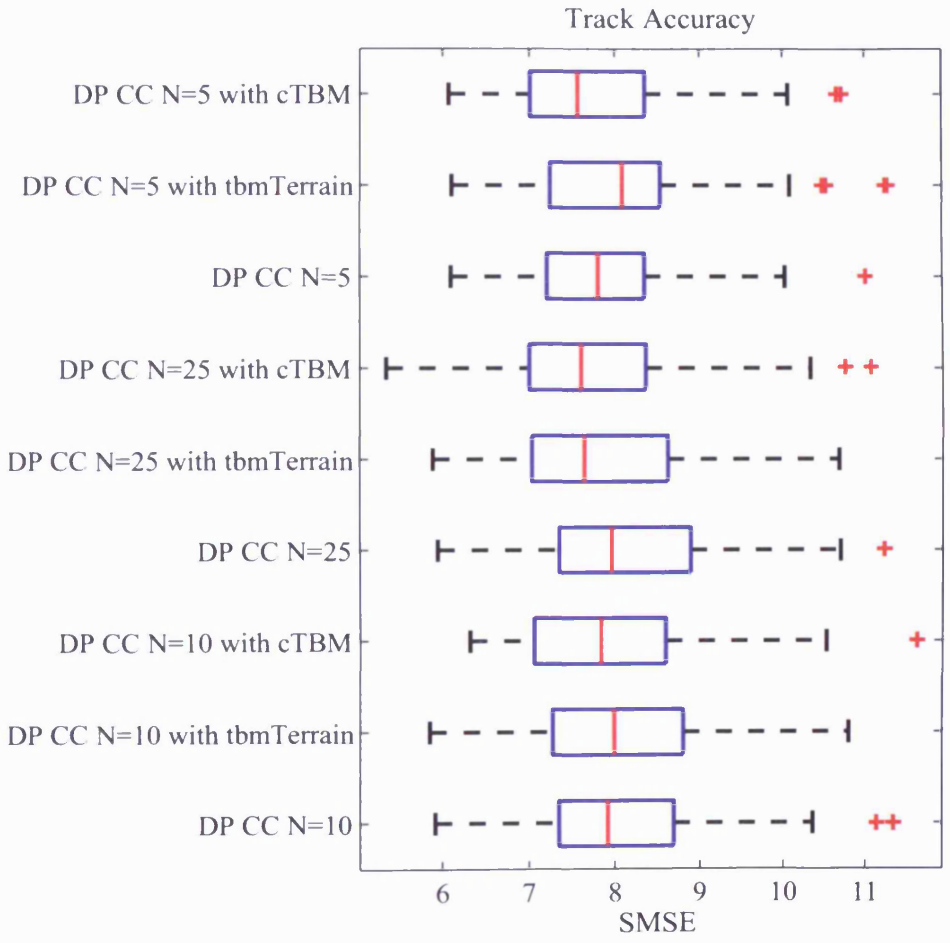


Figure 5.15: A comparison of track accuracies for the modified Scenario A. Each box plot is created from a set of 100 Monte Carlo trials.

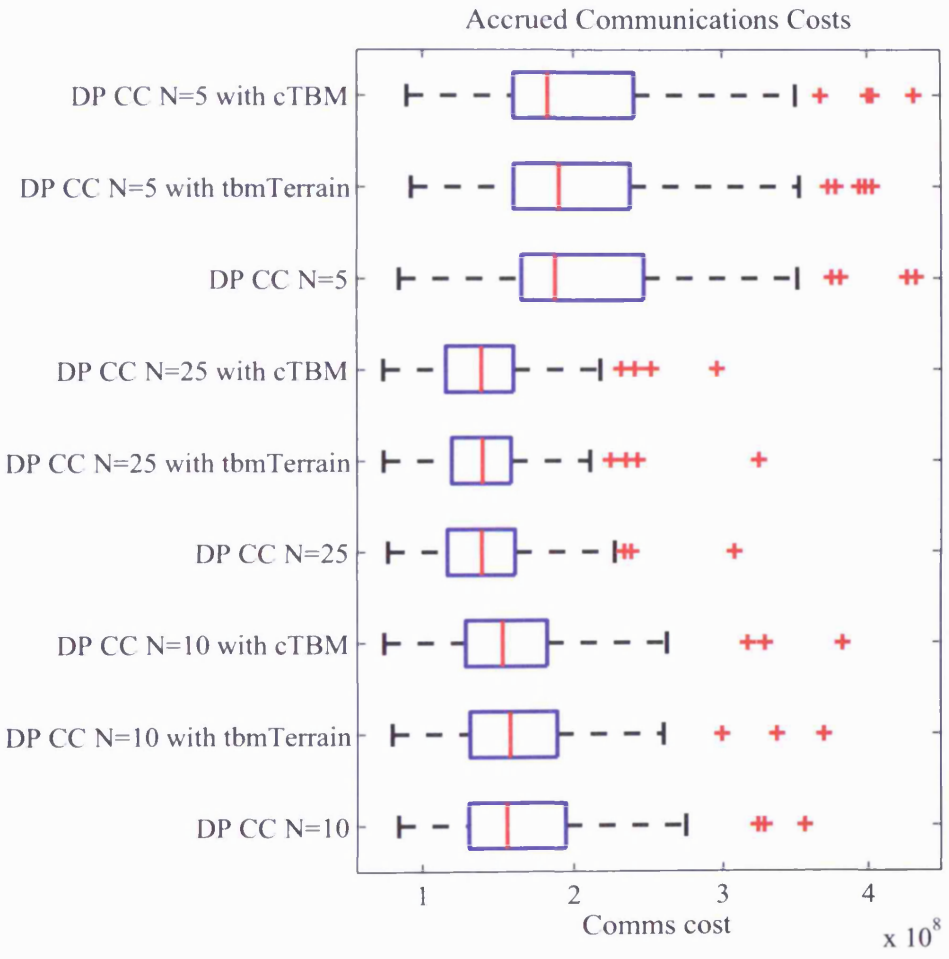


Figure 5.16: Accrued communications costs for the modified Scenario A. Each box plot is created from a set of 100 Monte Carlo trials.

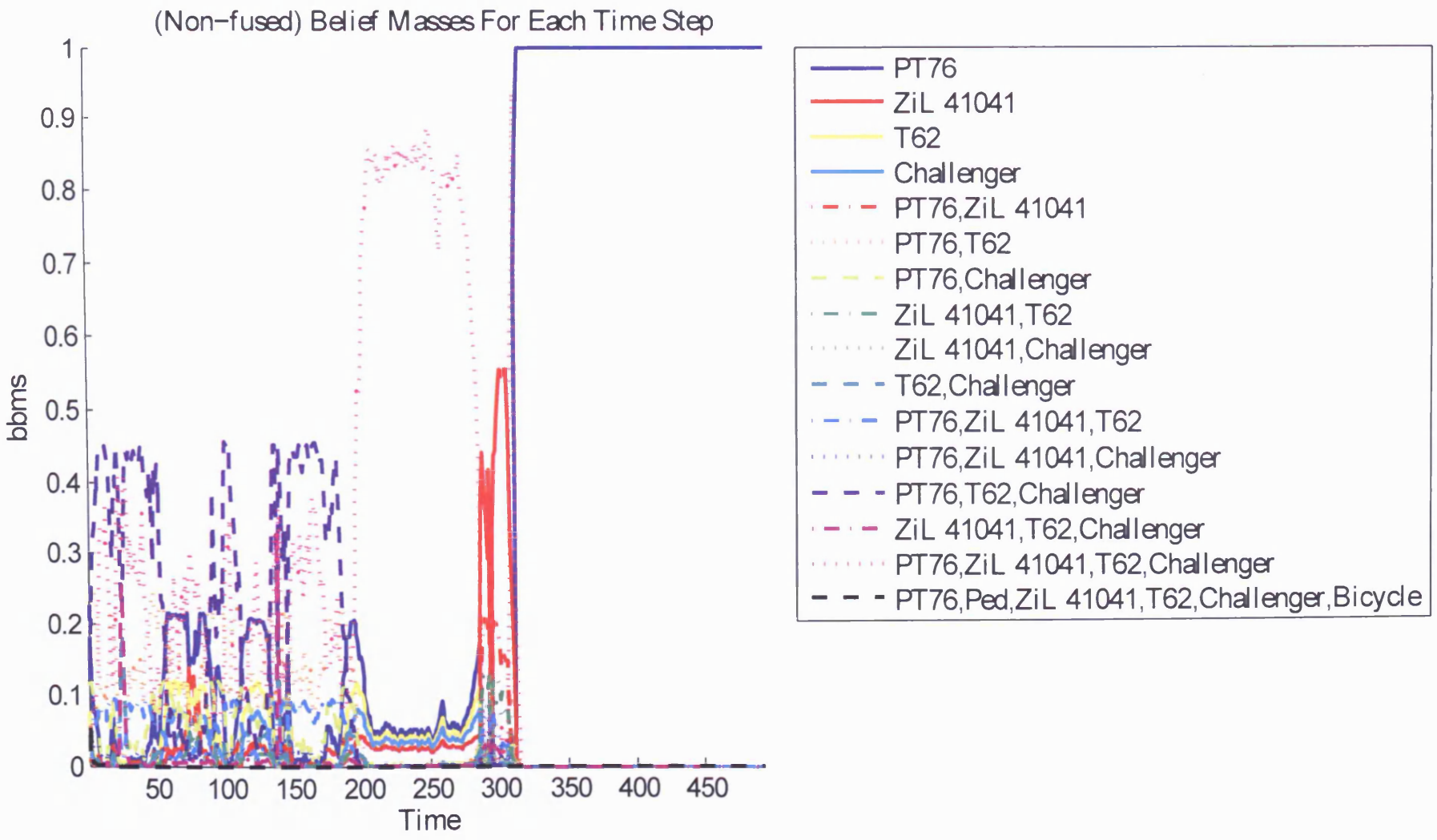


Figure 5.17: Non-fused belief masses for a typical modified Scenario A simulation using our new approach.

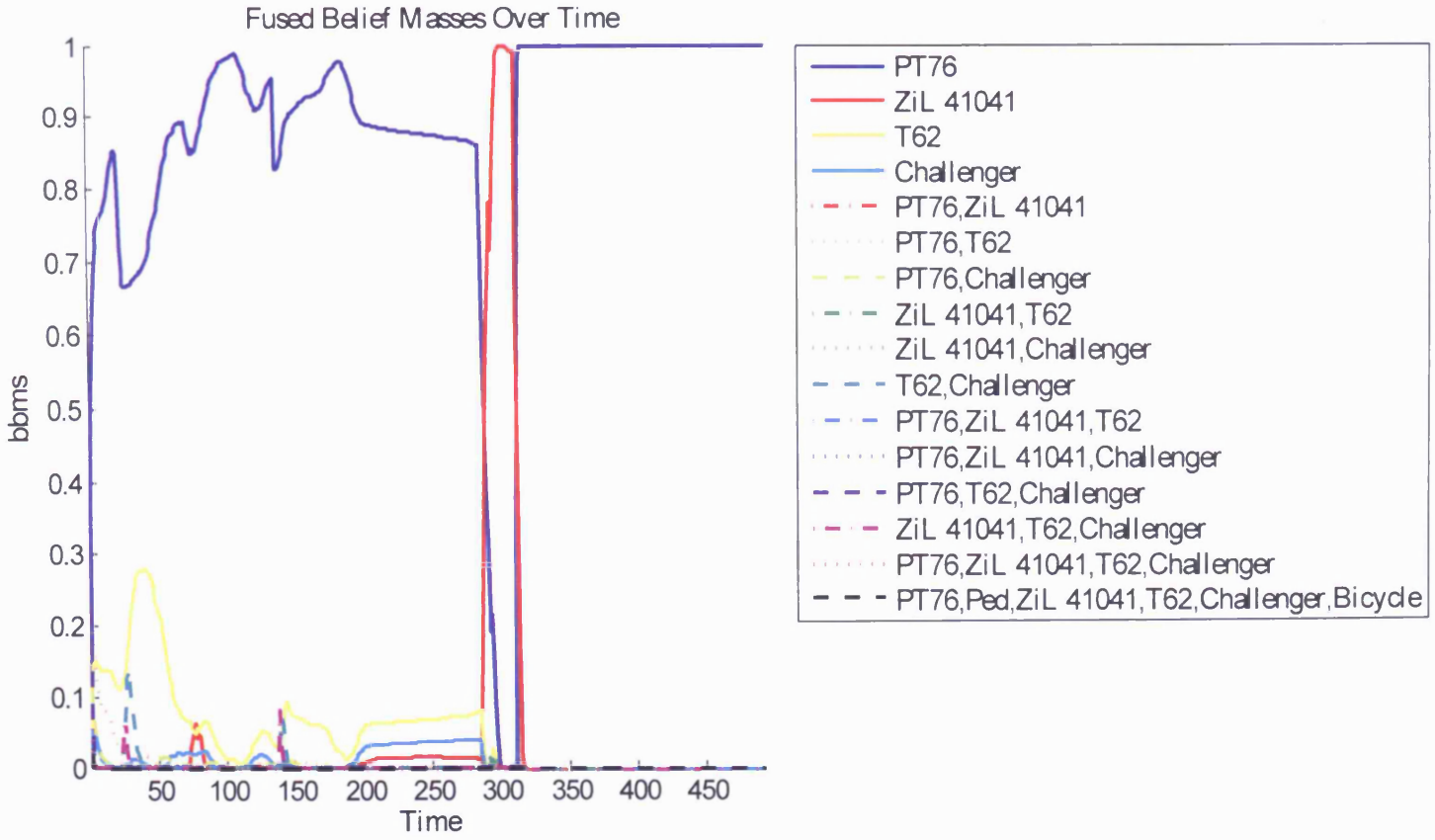


Figure 5.18: Fused belief masses for a typical modified Scenario A simulation using our new approach.

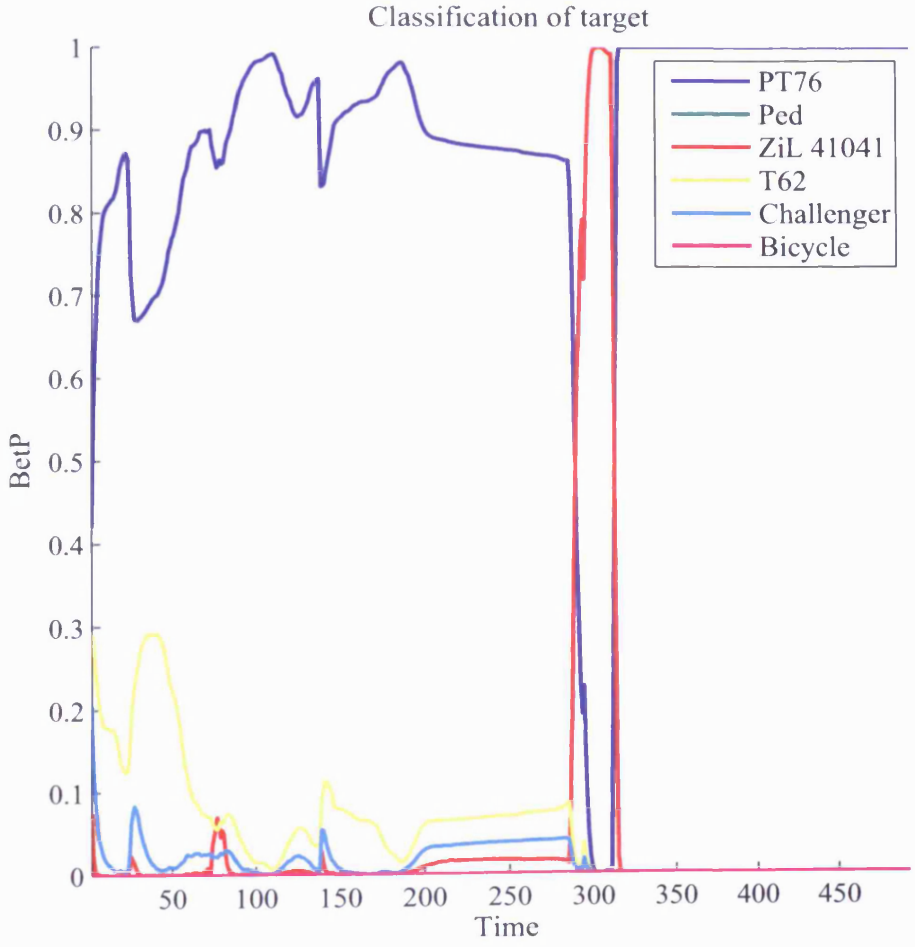


Figure 5.19: *BetP* for a typical modified Scenario A simulation using our new approach.

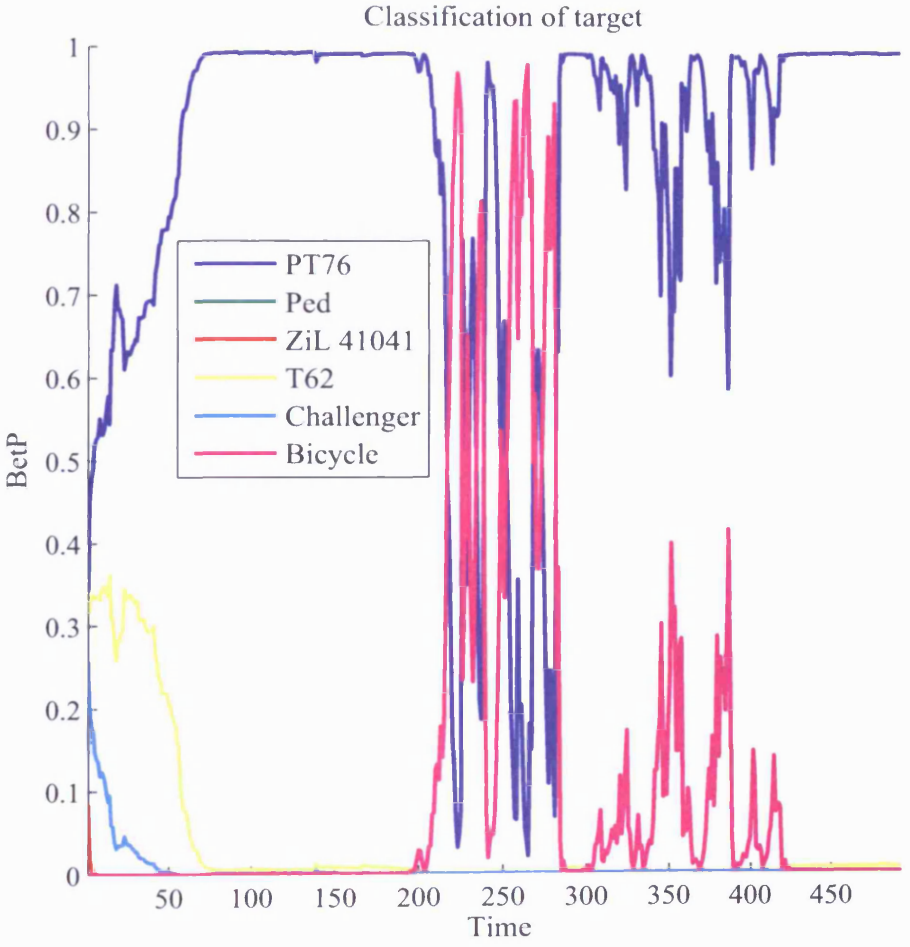


Figure 5.20: *BetP* for a typical modified Scenario A simulation using the approach of Chapter 4.

5.4.2 Scenario B

In Scenario B, a car travels along two sections of road and around a roundabout between the sections of road in a 898m by 576m region (See Figure 5.21). The target remains on road at all times and it is assumed that it is the only terrain type that it can travel on unhindered (See Table 5.2). 20 sensors are used in each simulation, with $R = 1.7$ and $a = 1000000$. Each Monte Carlo trial lasts 301 time steps. For tbmTerrain trials, $\iota = 20$, $\mathcal{W} = 7$, and $\rho = 1.15$. These parameters were determined experimentally.

A summary comparison can be found in Figure 5.22. The tracking performance, classification performance, and communications cost accrual of each approach can be found in Figures 5.24, 5.23, and 5.25 respectively. There is a substantial improvement in classification accuracy; this is likely to be due to the use of more realistic prior probabilities compared to the Gaussian priors used in Chapter 4.

There is no significant difference in tracking performance between approaches with the same horizon length — it is expected that this is for similar reasons to that of the results in Scenario A. As with Scenario A, a result of the tracking performance is that sensor usage is similar for all simulations with the same horizon length (See Figure 5.26), and hence communications cost are also similar.

The typical behaviour of the TBM with our new approach for Scenario B can be found in Figures 5.27–5.29. Figure 5.27 shows the non-fused *bbas* for each time step. The fused *bbas* are shown in Figure 5.28, and the resulting *BetP* for each time step can be found in Figure 5.29. Comparing the classification output from our new approach with the output of the approach taken in Chapter 4 (See Figure 5.30) it is clear that the classification from our new approach is more decisive and robust — in this particular case this is mainly due to only assigning masses to feasible classes.

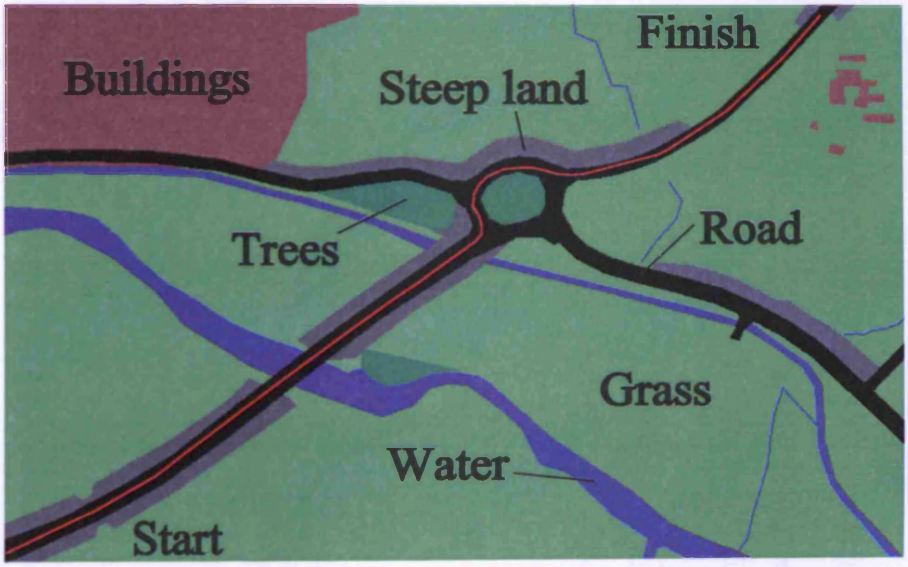


Figure 5.21: Target trajectory and terrain for Scenario B. Annotations have been added to the terrain map. The red line is the target trajectory.

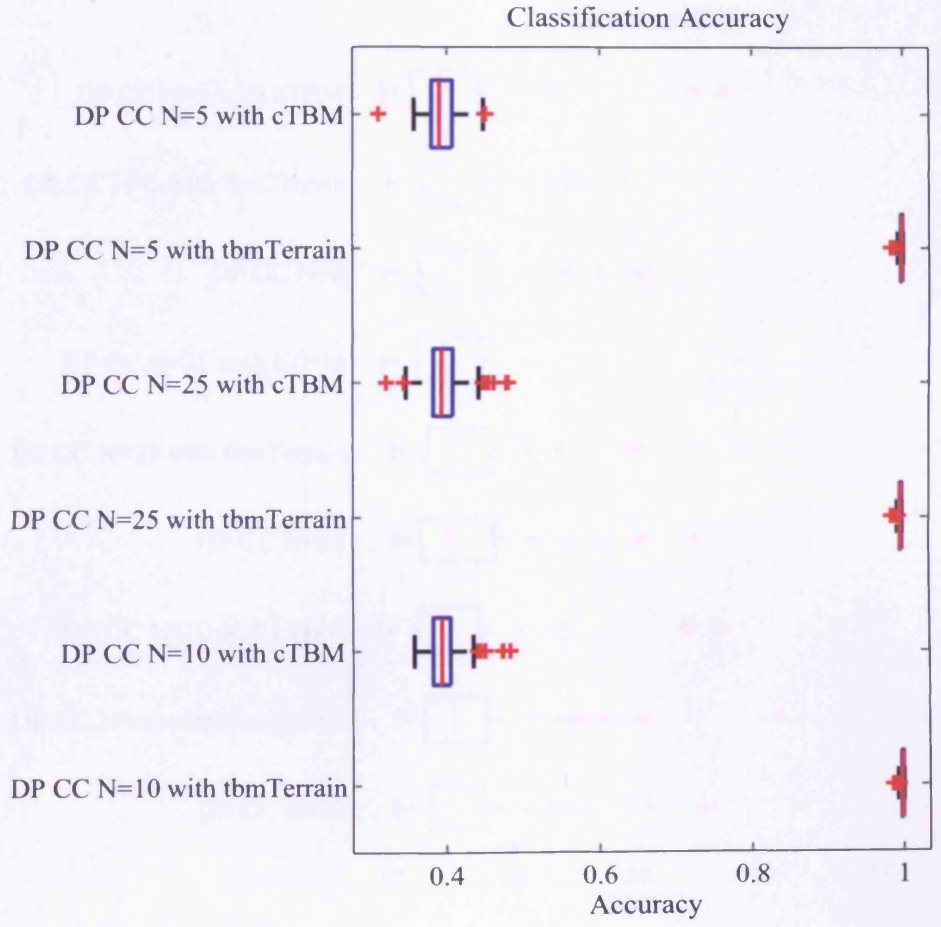


Figure 5.23: Classification accuracies for Scenario B. Each box plot is created from the mean value of $BetP(A)$ over time for each simulation.

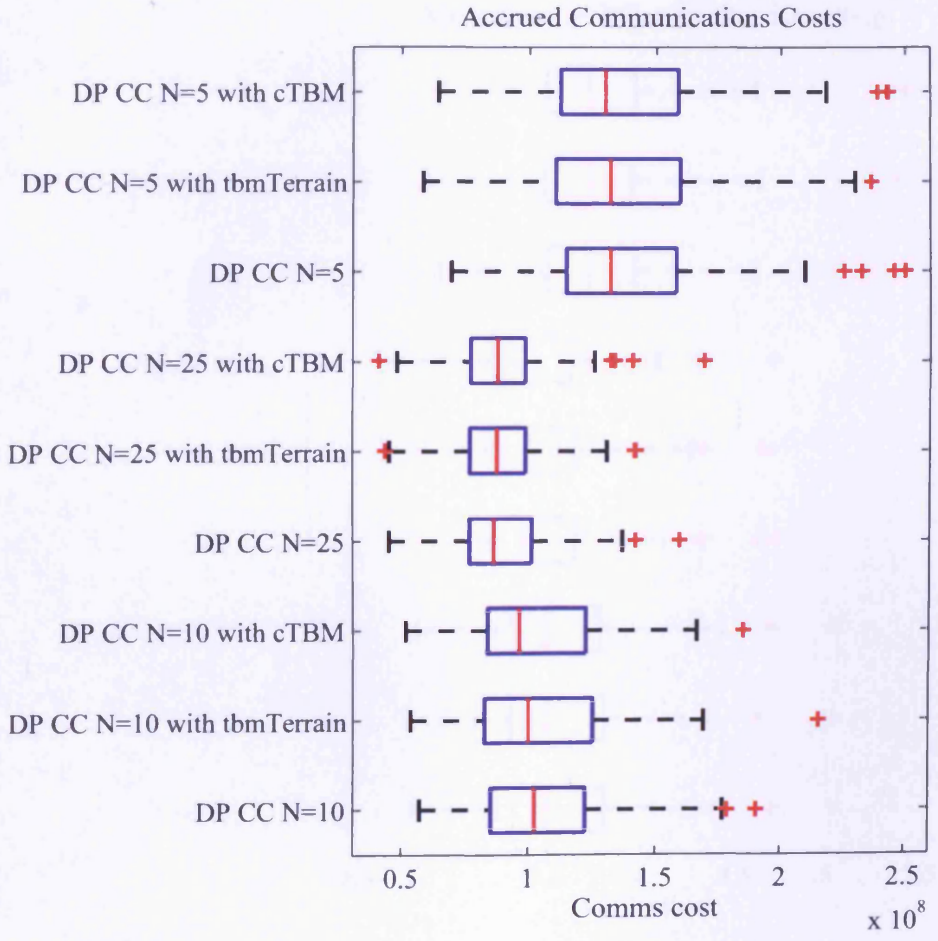


Figure 5.25: Accrued communications costs for Scenario B. Each box plot is created from a set of 100 Monte Carlo trials.

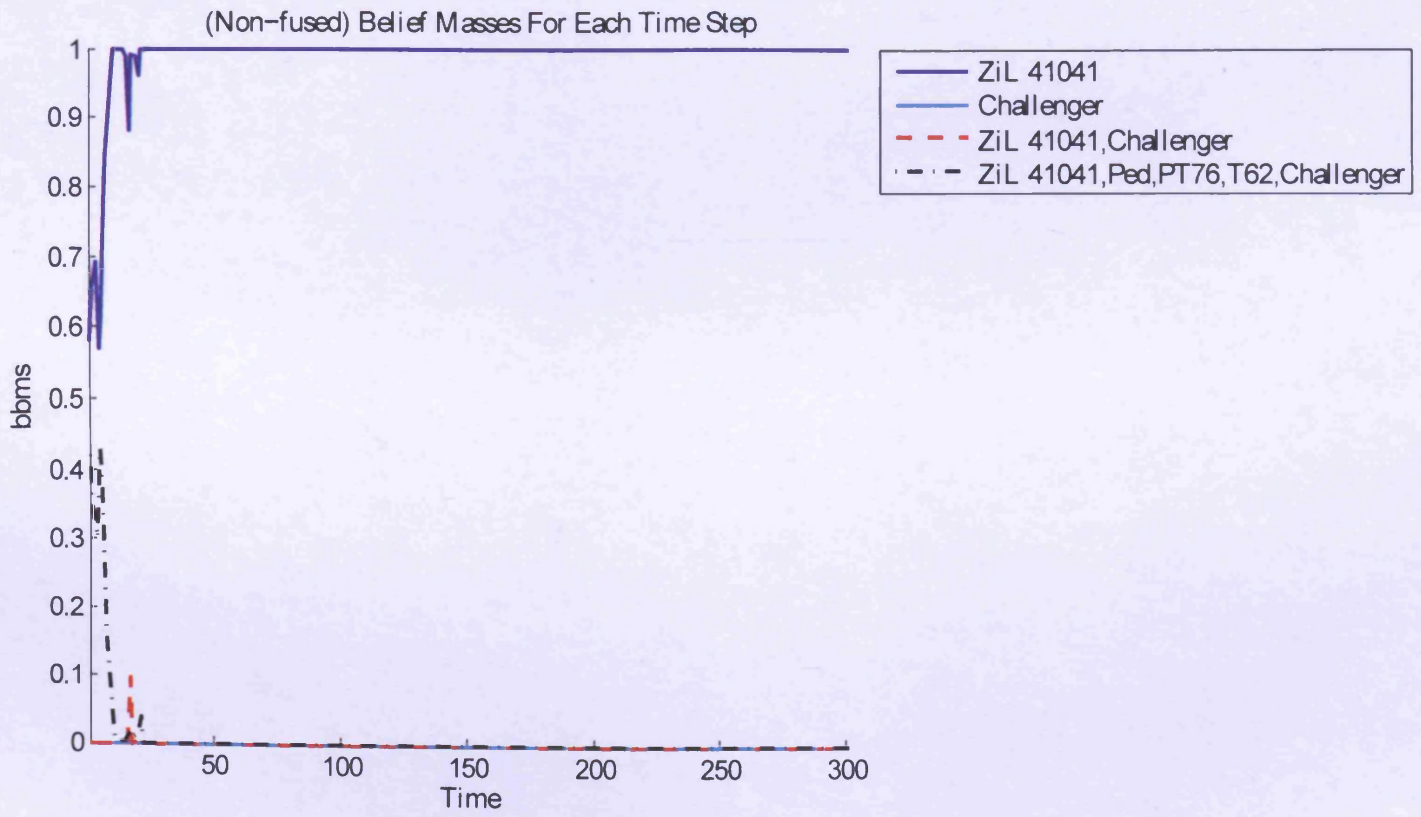


Figure 5.27: Non-fused belief masses for a typical Scenario B simulation using our new approach.

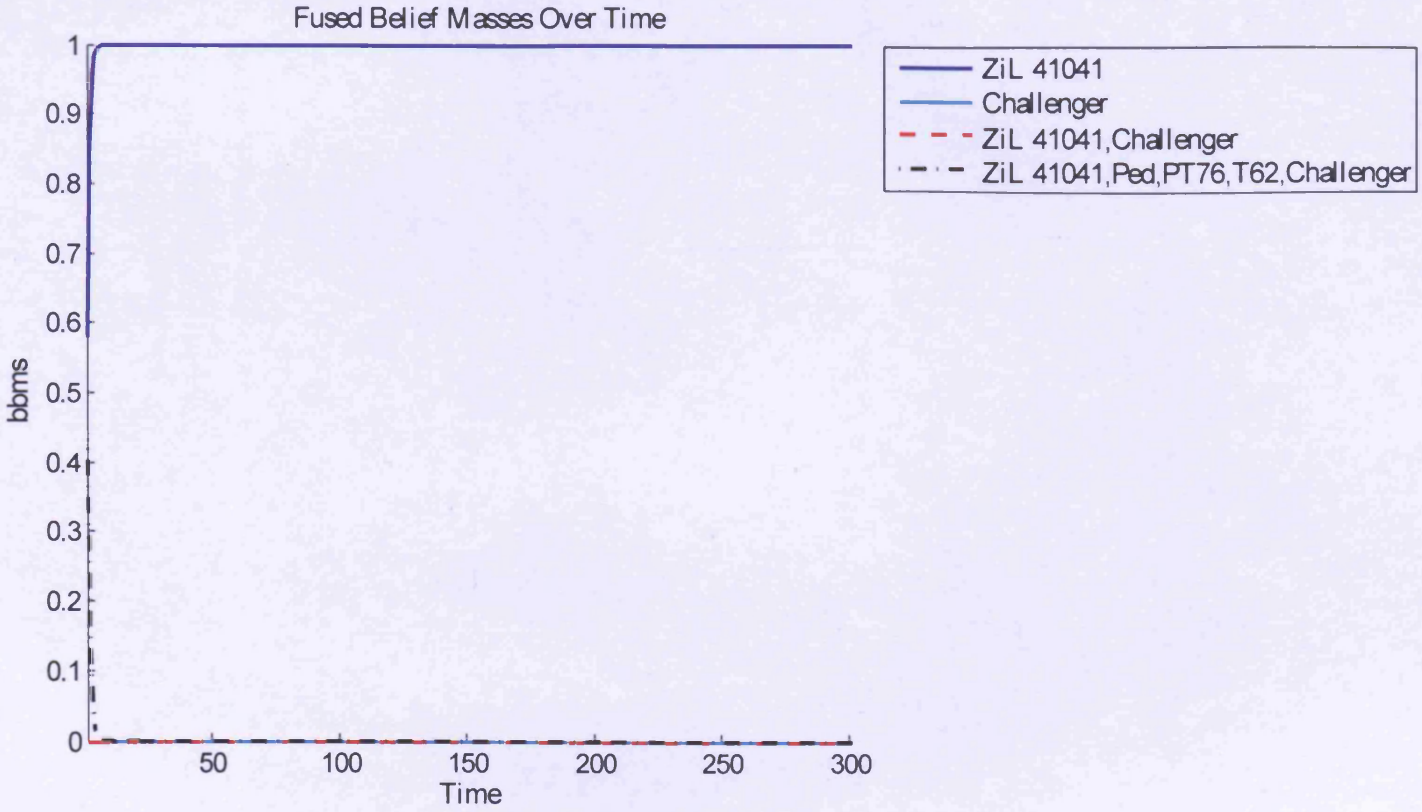


Figure 5.28: Fused belief masses for a typical Scenario B simulation using our new approach.

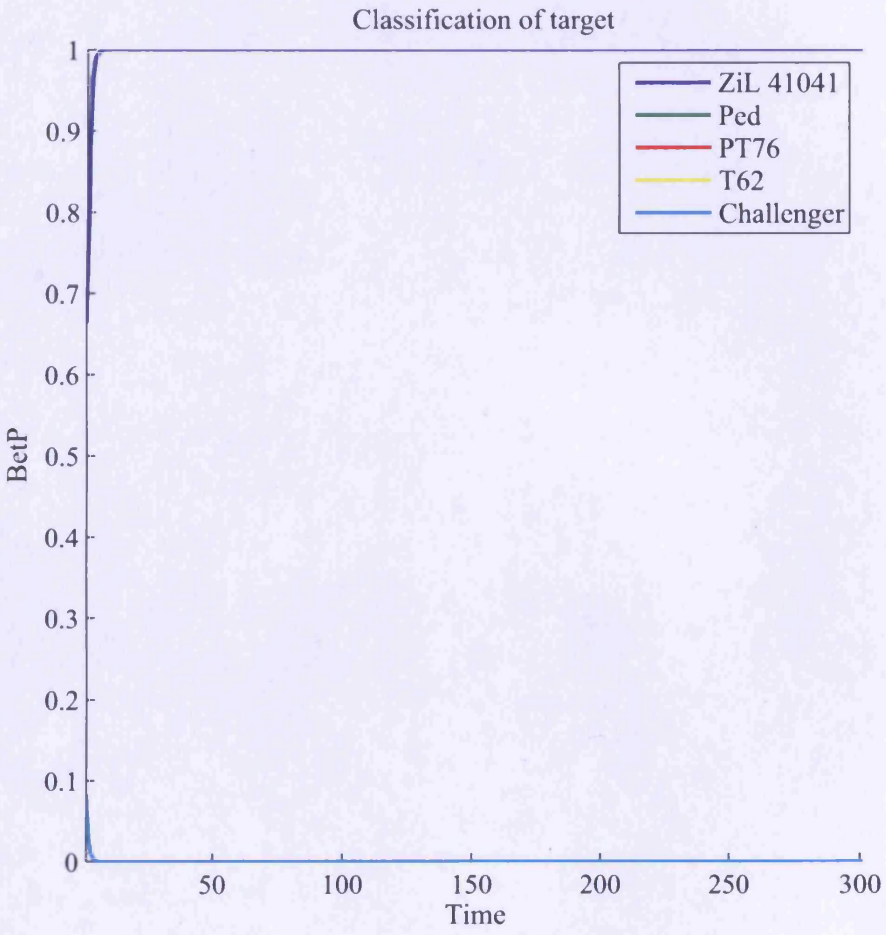


Figure 5.29: *BetP* for a typical Scenario B simulation using our new approach.

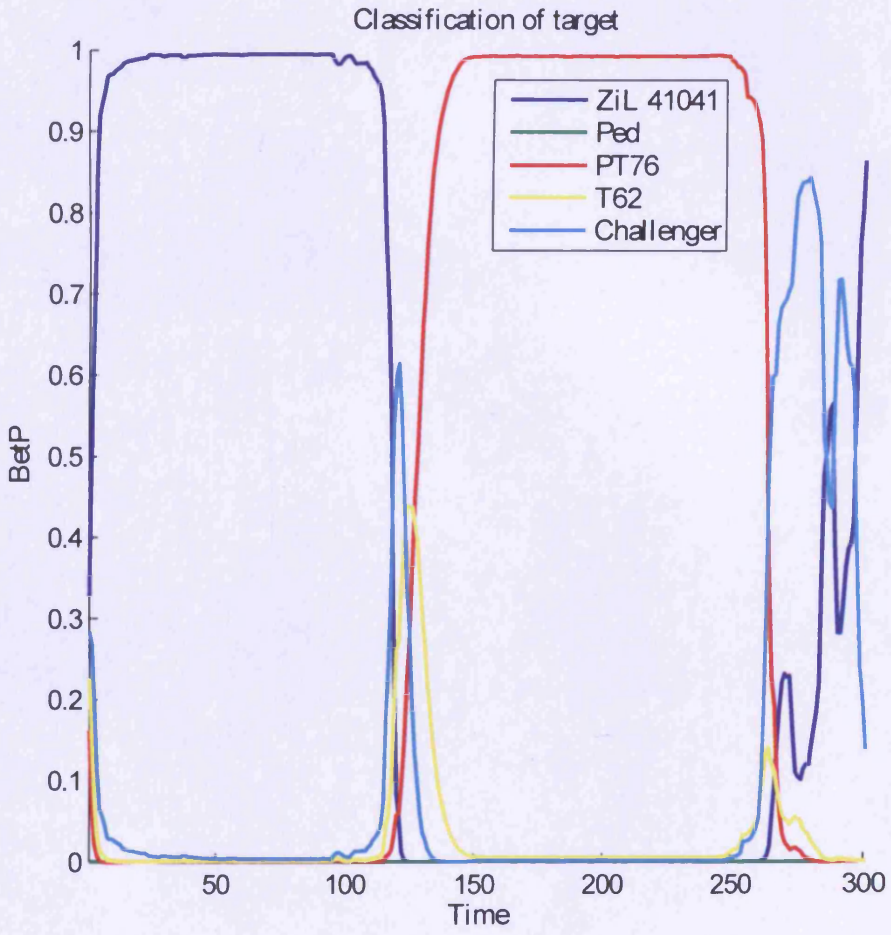


Figure 5.30: *BetP* for a typical Scenario B simulation using the approach of Chapter 4.

5.4.3 Scenario C

Scenario C consists of a main battle tank, travelling along a road and then travelling off-road in a 541m by 460m region (See Figure 5.31). 30 sensors are used in each simulation, with $R = 2.25$ and $a = 1000000$. Each Monte Carlo trial runs for 218 time steps. The tbmTerrain parameters are $\iota = 25$, $\mathcal{W} = 8$, and $\rho = 1.15$. These parameters were determined experimentally.

Figure 5.32 shows a performance summary. A comparison between the classification performance of our new approach and that of Chapter 4 is shown in Figure 5.33 and in Tables 5.7 and 5.8. The tracking performance of each approach is compared in Figure 5.34. Communications cost usage and sensor utilisation can be found in Figures 5.35 and 5.36 respectively. As with Scenarios A and B, the tracking performance is similar for all three approaches, resulting in similar sensor usage and hence similar communications costs.

From the results shown it is clear that there is a clear improvement in classification accuracy. There is no significant difference between the approaches for target tracking performance; like Scenarios A and B, it appears that this is an effect of a lack of significant improvement in tracking performance.

The typical behaviour of the TBM for Scenario C using our new approach can be found in Figures 5.37–5.39. Figure 5.37 shows the non-fused *bbas* for each time step. The fused *bbas* are shown in Figure 5.38, and the resulting *BetP* for each time step can be found in Figure 5.39.

With tbmTerrain, soon after the initialisation period, ι, most of the belief assigned from the estimate at each time step (The non-fused masses) is assigned to subsets of Ω containing only the main battle tank, the car, or both. This is due to a combination of the prior probabilities for each class and the mechanism that ignores classes that are no longer feasible. At approximately time step 115, belief is only assigned to the main battle tank — this is because the car is not capable of travelling quickly off-road. This can be seen in the partial confusion matrix for the tbmTerrain approach (Table 5.8), where for the majority of the time, the target is classified correctly.

The classification output for a typical scenario using the approach of Chapter 4 is very different. In Figure 5.40 it can be seen that the classification is inaccurate for a substantial length of time and changes in a reasonably smooth manner. The non-fused masses at each time step reflect which target class has the highest likelihood for the estimated target speed, as the target changes speed gradually so does the assignment of masses, and hence *BetP* changes gradually. Since the approach of Chapter 4 does not take into account past behaviour or terrain information it cannot assign masses in a more sensible manner like tbmTerrain does. The partial confusion matrix (Table 5.7)

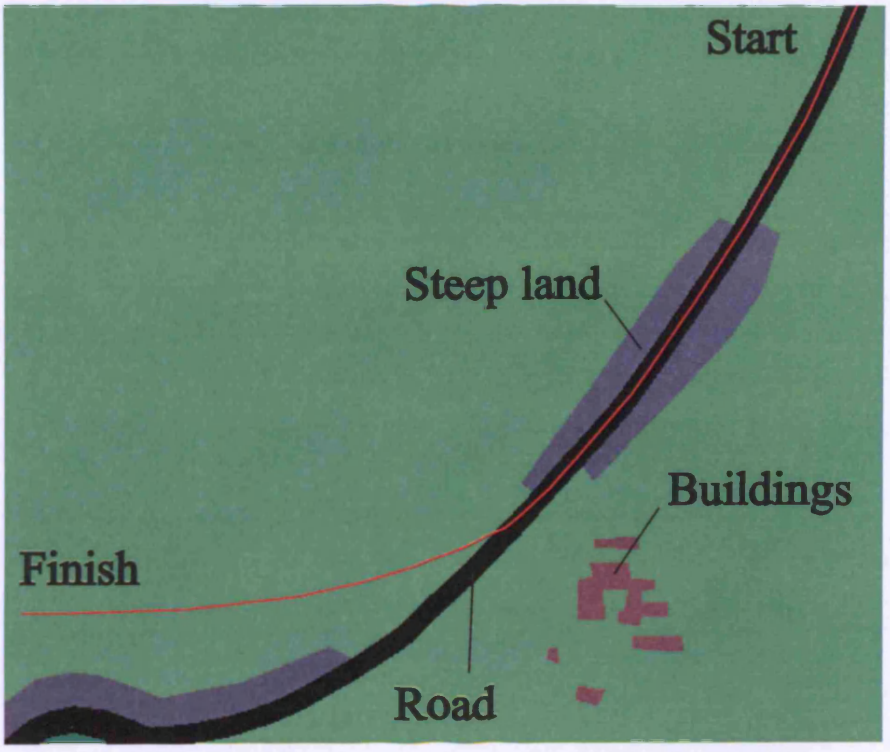


Figure 5.31: Target trajectory and terrain for Scenario C. Annotations have been added to the terrain map. The red line is the target trajectory.

highlights the poor classification performance of the this approach, where the output appears to be very indecisive.

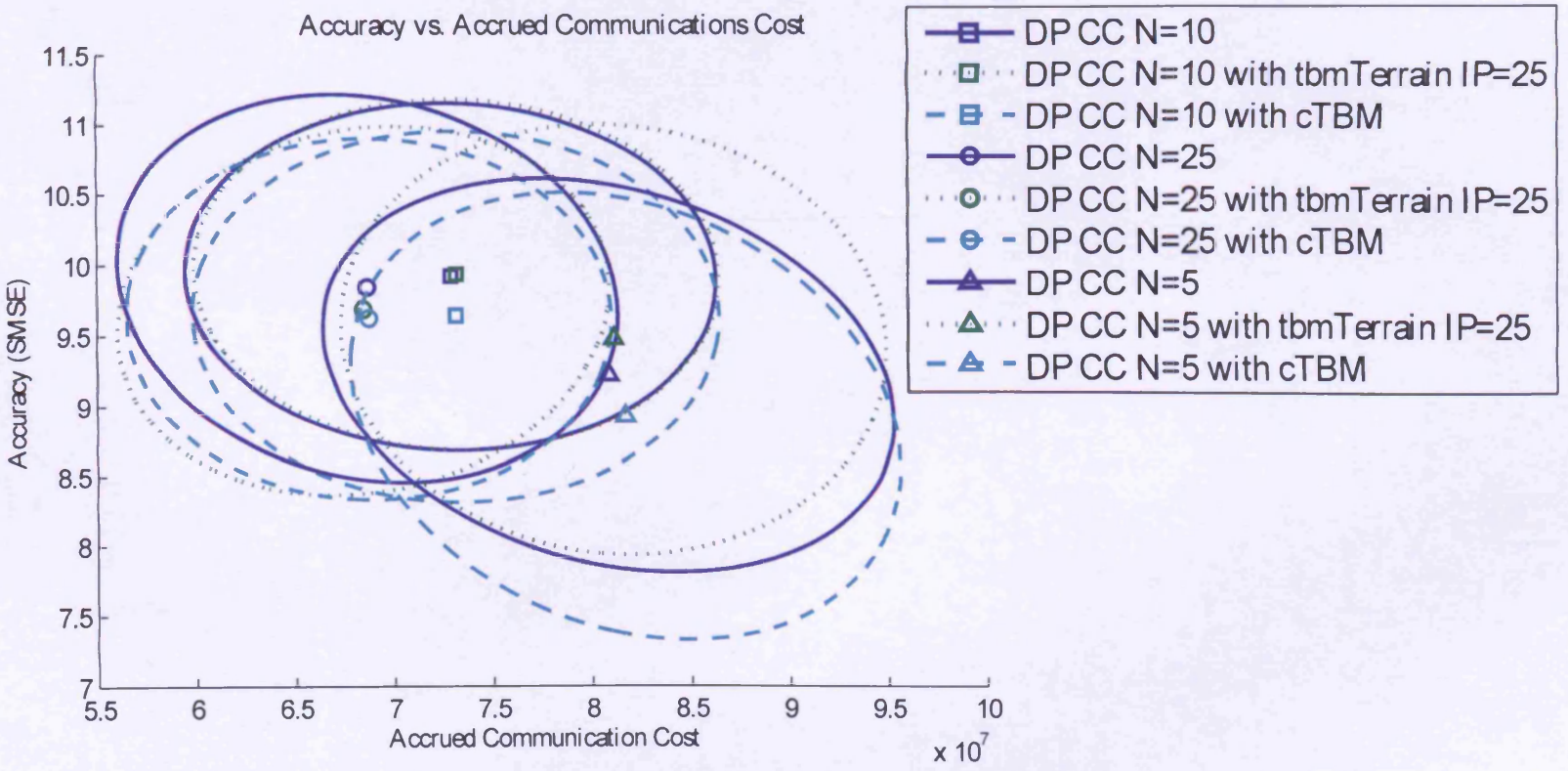


Figure 5.32: An overview of track accuracy and accrued communications costs for Scenario C.

		Predicted				
		Pedestrian	Amphibious Light Tank	Light Tank	Main Battle Tank	Car
Actual	Pedestrian	0	0	0	0	0
	Amphibious Light Tank	0	0	0	0	0
	Light Tank	0	0	0	0	0
	Main Battle Tank	0	248	14636	18581	31935
	Car	0	0	0	0	0

Table 5.7: Partial confusion matrix for Scenario C using the JTAC approach of Chapter 4.

		Predicted				
		Pedestrian	Amphibious Light Tank	Light Tank	Main Battle Tank	Car
Actual	Pedestrian	0	0	0	0	0
	Amphibious Light Tank	0	0	0	0	0
	Light Tank	0	0	0	0	0
	Main Battle Tank	0	2096	794	59663	2847
	Car	0	0	0	0	0

Table 5.8: Partial confusion matrix for Scenario C using the tbmTerrain approach.

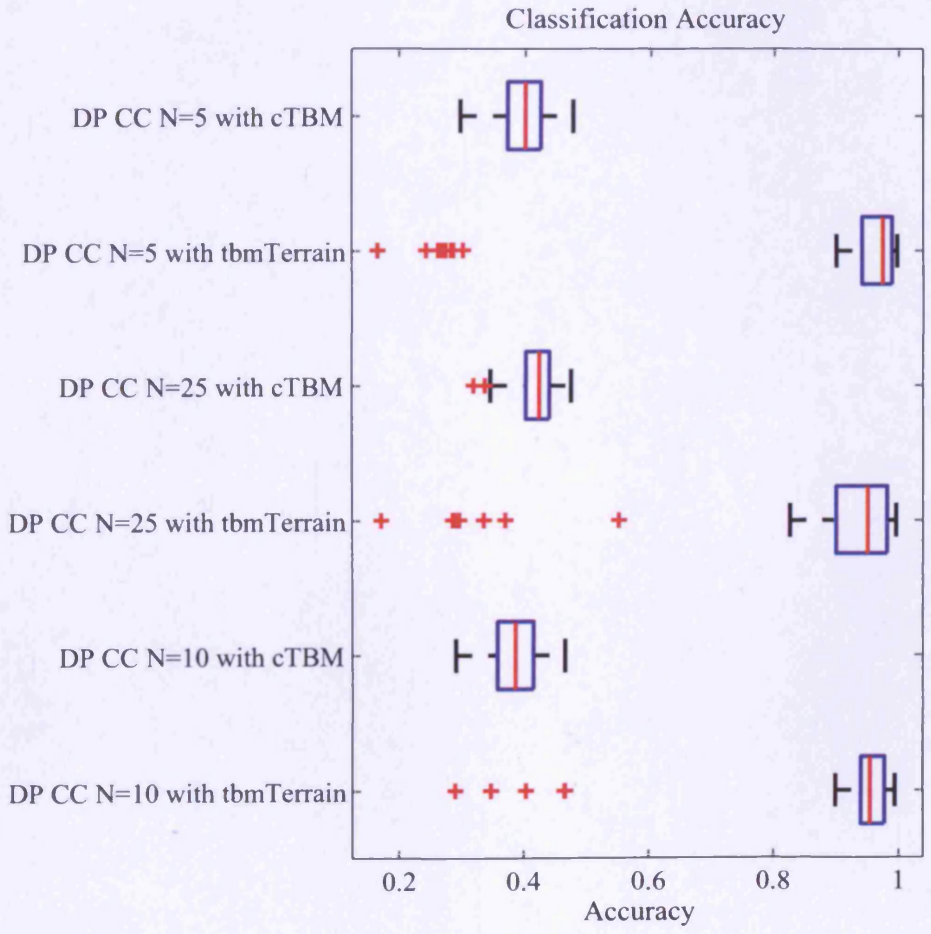


Figure 5.33: Classification accuracies for Scenario C. Each box plot is created from the mean value of $BetP(A)$ over time for each simulation.

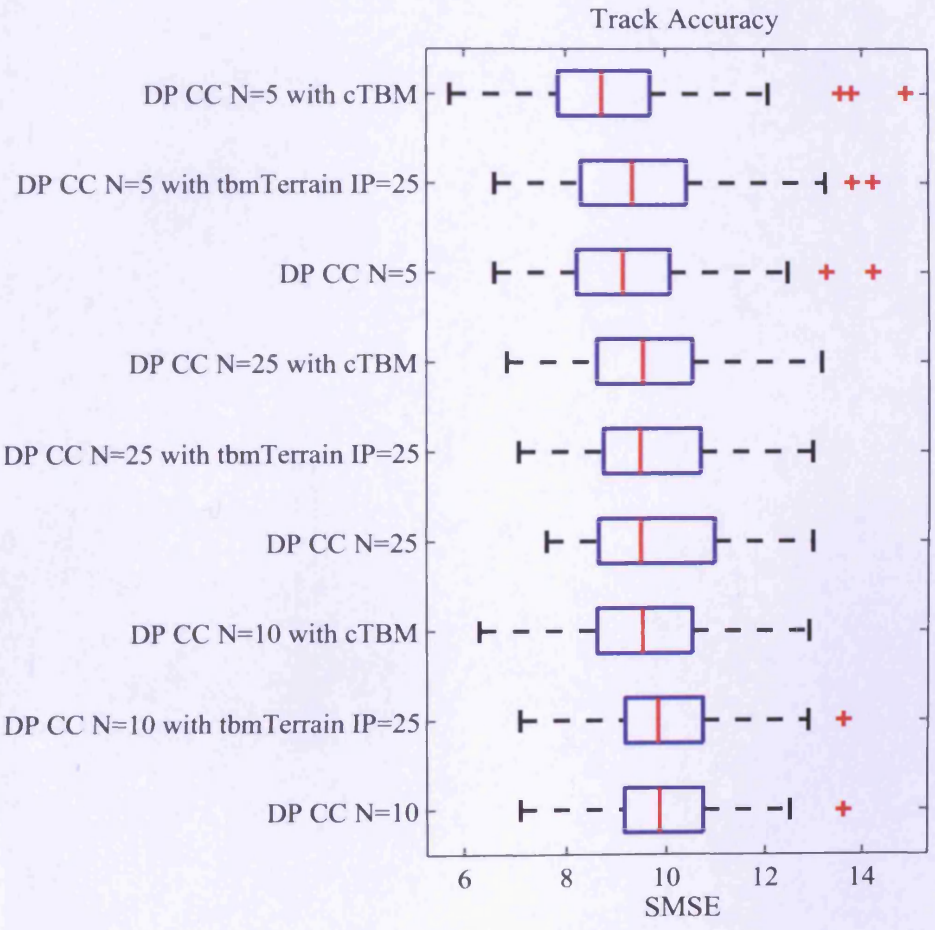


Figure 5.34: A comparison of track accuracies for Scenario C. Each box plot is created from a set of 100 Monte Carlo trials.

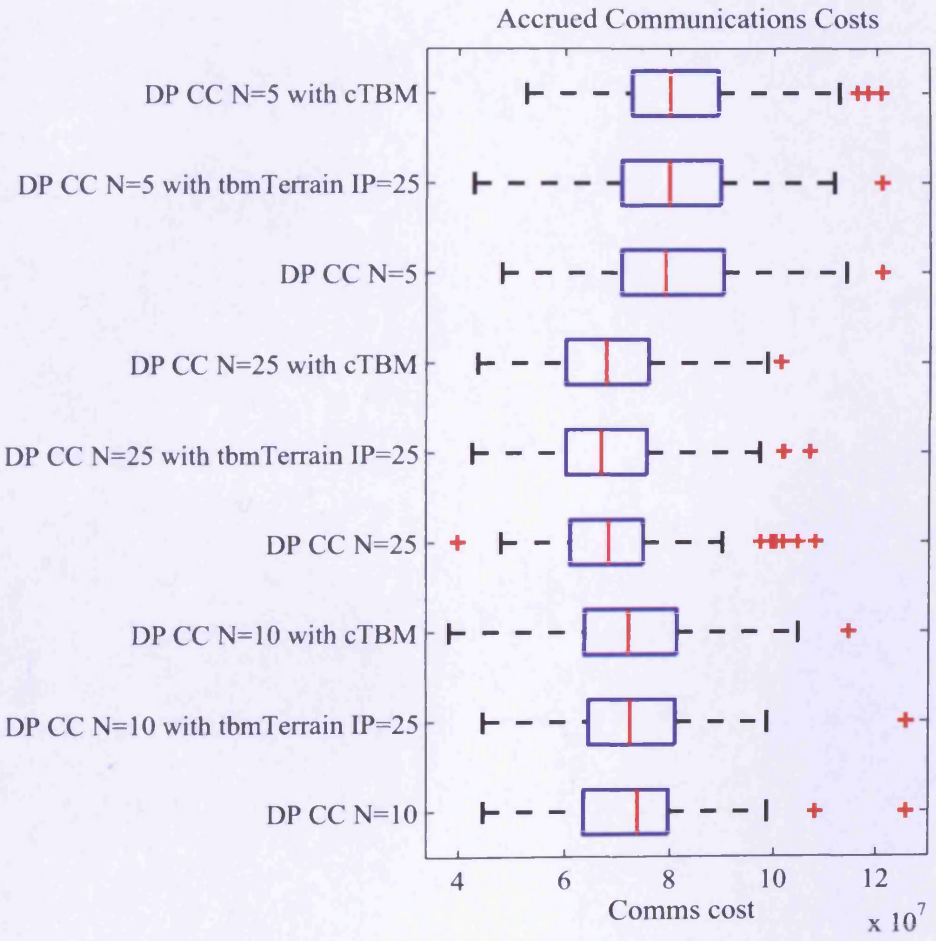


Figure 5.35: Accrued communications costs for Scenario C. Each box plot is created from a set of 100 Monte Carlo trials.

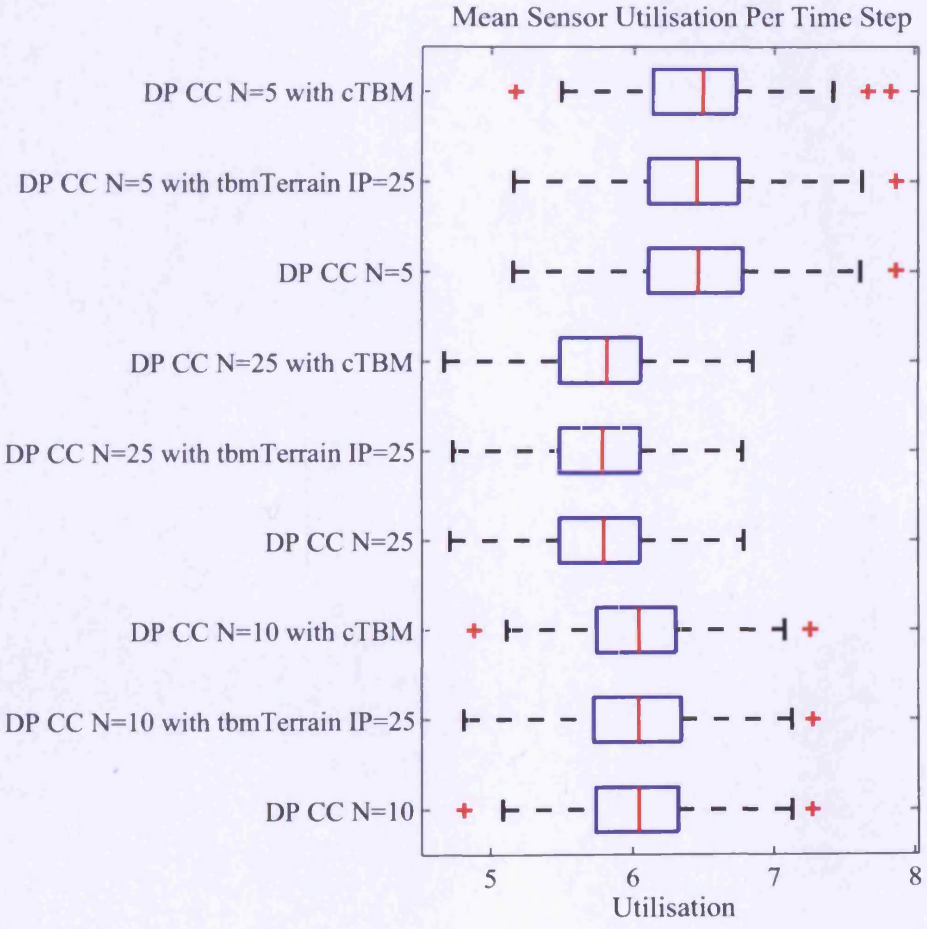


Figure 5.36: Mean number of sensors used per time step for Scenario C. Each box plot is created from a set of 100 Monte Carlo trials.

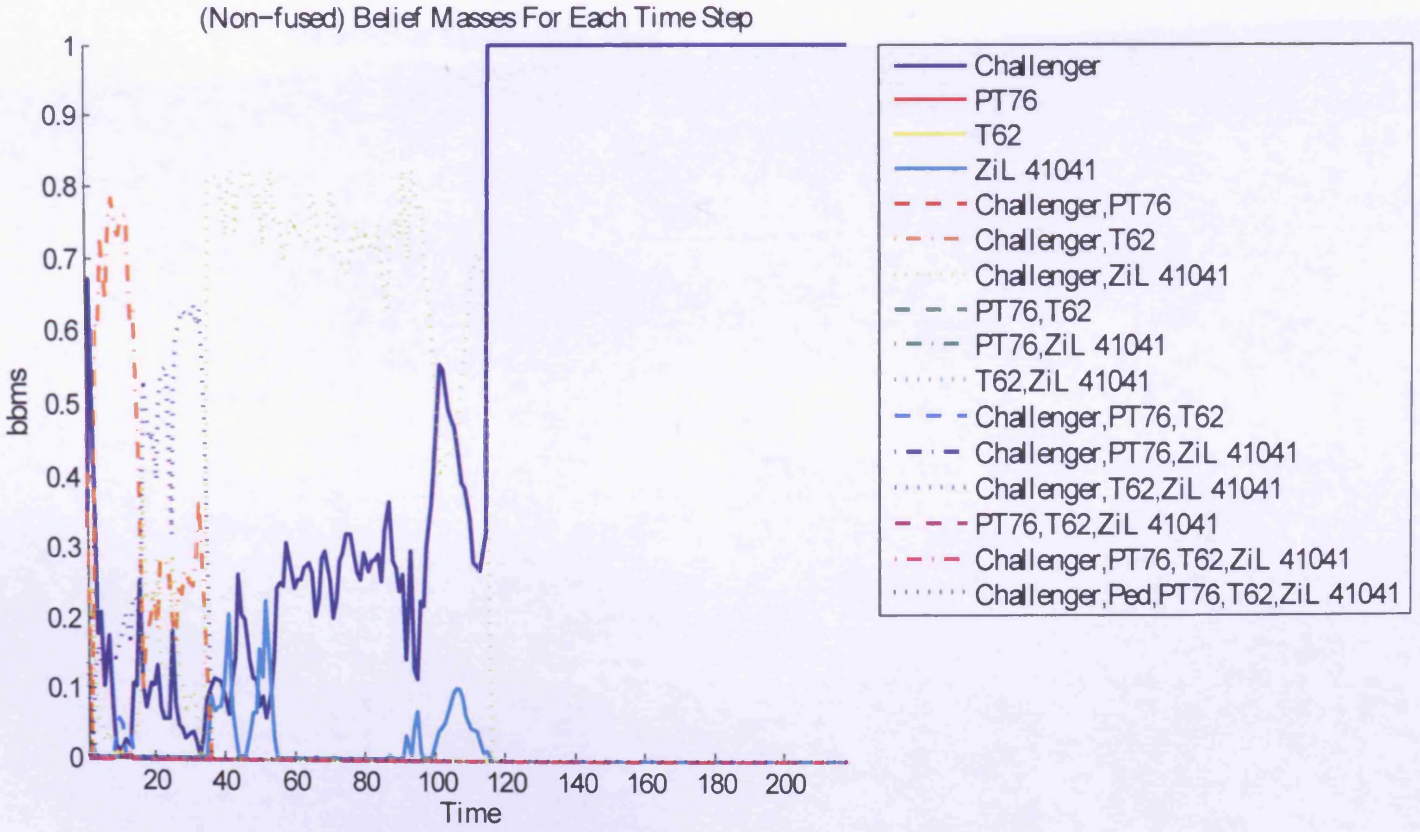


Figure 5.37: Non-fused belief masses for a typical Scenario C simulation using our new approach.

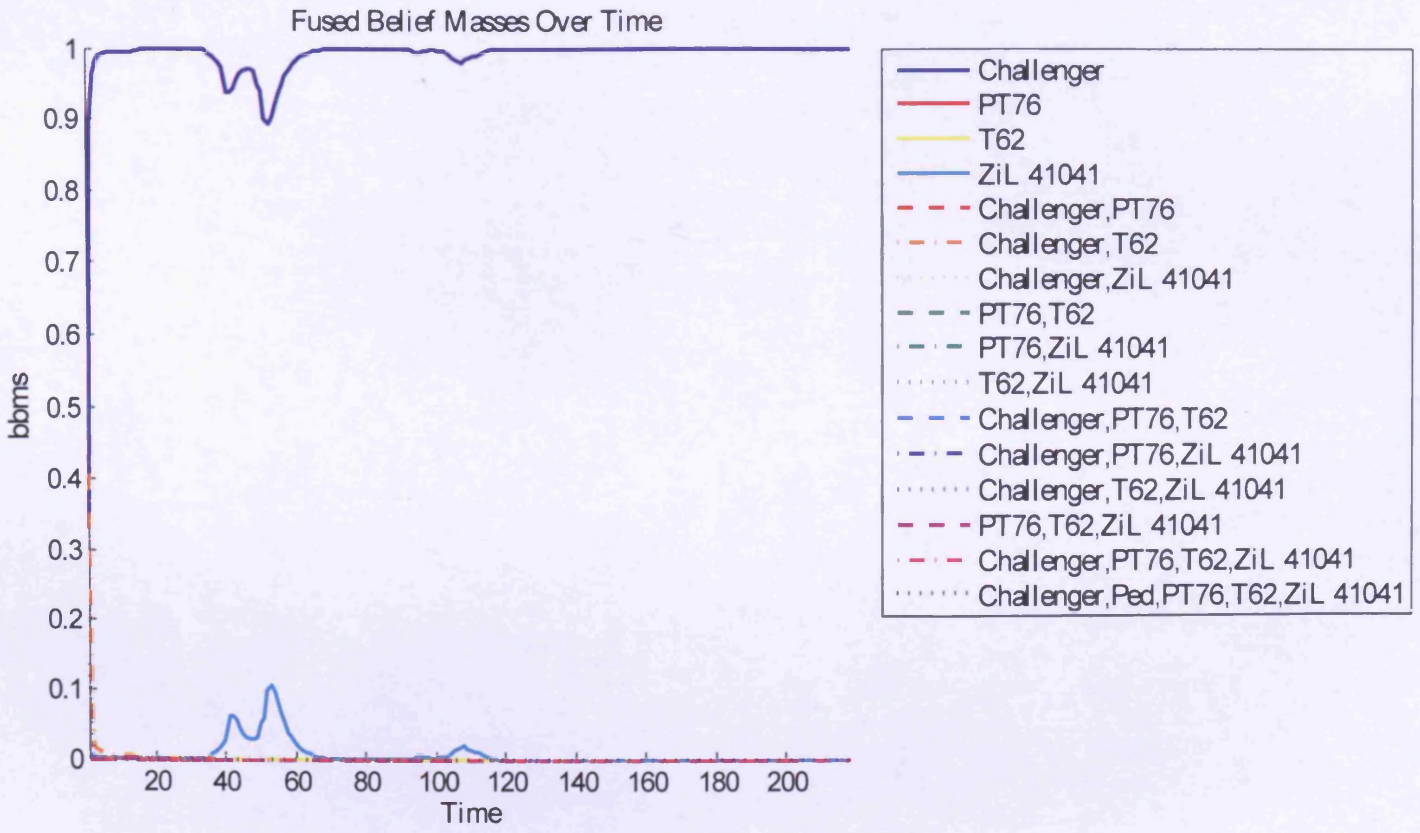


Figure 5.38: Fused belief masses for a typical Scenario C simulation using our new approach.

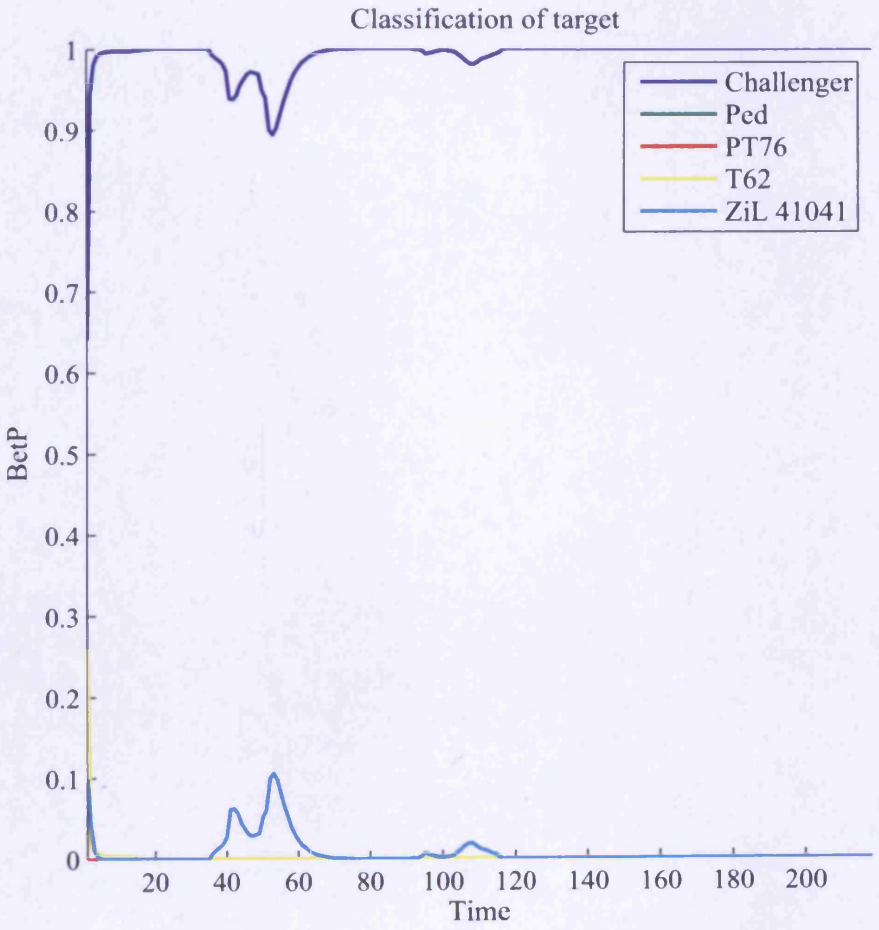


Figure 5.39: *BetP* for a typical Scenario C simulation using our new approach.

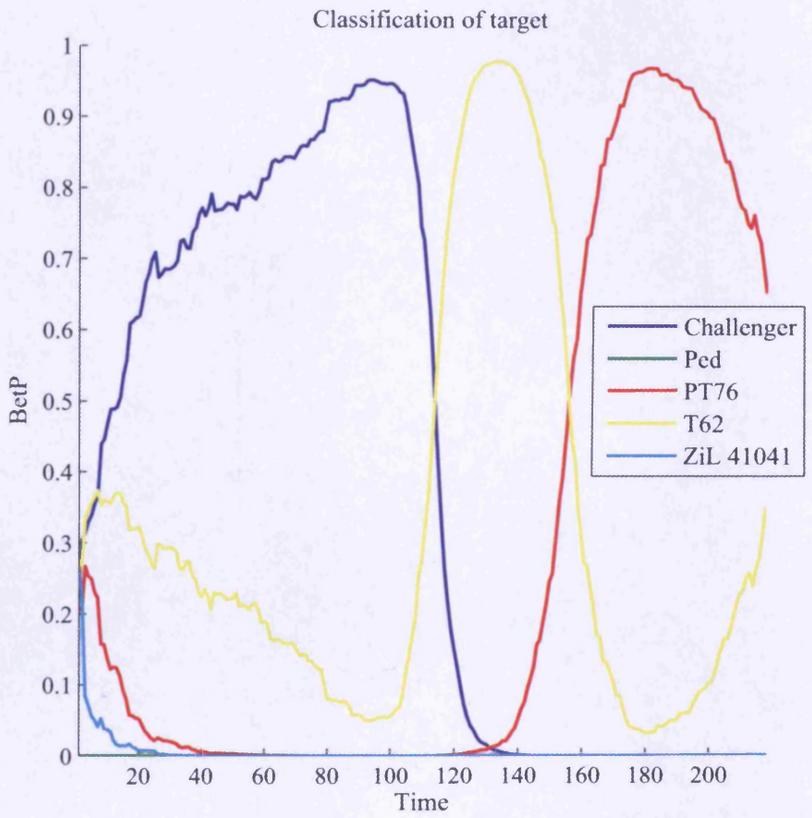


Figure 5.40: *BetP* for a typical Scenario C simulation using the approach of Chapter 4.

5.5 Conclusions

In this chapter we have extended the work of Chapter 4 to produce a more sophisticated WSN JTAC algorithm. Most of the work presented in this chapter has focused on improving classification performance — which includes the use of terrain information in the classification process. Our new approach to WSN JTAC, which takes place within the framework of Chapter 3, has been evaluated along with the approach of Chapter 4 and the tracking-only approach of Williams et al. (2007). Details of the improvements have been presented in Section 5.3. A condensed version of this chapter has been published (Roberts et al., 2010).

This chapter has presented various novel methods to improve classification with the TBM. One of which is the use of terrain information related to how it restricts each target class's movement. Another is the use of an elliptical area of the terrain map to account for position uncertainty and the subsequent weighted combination of conditional plausibilities related to the terrain coverage within the ellipse. We have also added a memory to the TBM to prevent it from assigning plausibility (and hence belief) to outcomes that earlier estimates have shown is not feasible.

A number of improvements have been utilised in this chapter including the use of more realistic scenarios, estimated target speed smoothing, and more realistic prior probabilities. The simplistic nature of the scenario used in Chapter 4 does *not* allow algorithms to be evaluated in such a way that may reflect real world performance; this chapter has used more realistic scenarios, with non-linear target trajectories affected by terrain and targets based on real vehicles. The estimated speed of the target has been smoothed to reduce the noisy estimate produced by the particle filter. More realistic prior probabilities have been used to reflect the more realistic nature of the targets used in scenarios of this chapter; previously Gaussian priors were used but these do not accurately reflect the varying speed of targets.

The results have shown a consistently good classification performance for our new approach; the increase in performance is due to the improvements presented in Section 5.3. From the tracking accuracy results shown in Section 5.4, it appears that the new method of feedback from the TBM to the particle filter does not improve tracking accuracy. A consequence of the poor feedback from the TBM to the particle filter is that the target tracking for all three approaches evaluated in this chapter perform have a similar performance (for the same horizon length), resulting in similar sensor utilisation levels and associated communications costs.

5.6 Future Work

There are a number of potential improvements that could be made to the work presented in this chapter. Firstly, the uncertainty of the target's position could be more effectively used than a simple uncertainty ellipse; since the ellipse represents a Gaussian distribution cut-off at a set threshold better results might be achieved by weighting the terrain according to how far it is from the centre of the ellipse. A more sophisticated option would be to not use an uncertainty ellipse, as this may not accurately model uncertainty if a Gaussian distribution cannot be used to sufficiently model the probability distribution of the target's position. Instead, a *bba* could be created for each particle, taking into account the terrain at the particle's position — however this would significantly increase computational costs.

An effective method of providing feedback from the TBM to the particle filter remains to be found, the poor performance of this step limits the overall performance of both the approach presented in this chapter and that of Chapter 4. Once a more effective method for feedback has been found, it will be possible to see if the reduction in the uncertainty of the kinematic state estimate reduces the sensor usage and associated communications costs.

As with the work presented in Chapter 4, we have limited ourselves to closed world scenarios; this prevents our approach from coping with target classes that are not within a pre-determined set. Ideally, our approach would use an open world but unfortunately due to amount of conflict between belief masses created at different time steps, more research is required before this can take place.

In the next chapter, we show how the framework, presented in Chapter 3, can be used as a tool to aid sensor deployment planning. The algorithms developed in this chapter and the previous chapter will be used in the next chapter to demonstrate how sensor deployment affects WSN performance.

Chapter 6

Planning Sensor Deployment for Joint Tracking and Classification

Unlike previous chapters, which focused on using the framework presented in Chapter 3 to experiment with JTAC approaches, this chapter focuses on using the framework as a tool to aid sensor deployment planning. We hypothesise that a WSN with sensors strategically positioned to take into account the terrain and therefore the likely target trajectories, will perform better than a WSN that uses randomly positioned sensors. An improvement in tracking performance will be seen, which will result in an improvement in classification performance.

In the basic scenario of Chapter 4, using a uniform random distribution for sensor deployment is intuitive — there is no prior knowledge of where the target will travel within the region of interest. The scenarios of Chapter 5 were different. The terrain combined with a knowledge of the potential targets (The frame of discernment) provides clues as to where the target may travel. For example, an area at the top of the terrain in Scenario A is steep land — a type of terrain that most of the targets cannot traverse. When planning the deployment of WSN nodes, performance gains can be achieved by taking this prior knowledge into account, positioning nodes in areas where the target is likely to be.

This change in usage of the aforementioned framework highlights its flexibility; the framework can be used both for experimenting with new JTAC algorithms and in *planning* WSN deployment. The use of this framework to aid the planning of sensor deployment could be of use to both military and civilian organisations — although some aspects of the framework would need to be improved to simulate more complex scenarios, for example, tracking multiple targets simultaneously. The use of simulations to experiment with sensor deployment provides an attractive option for quick, lower cost investigations compared to real world investigations. The framework cannot be used to

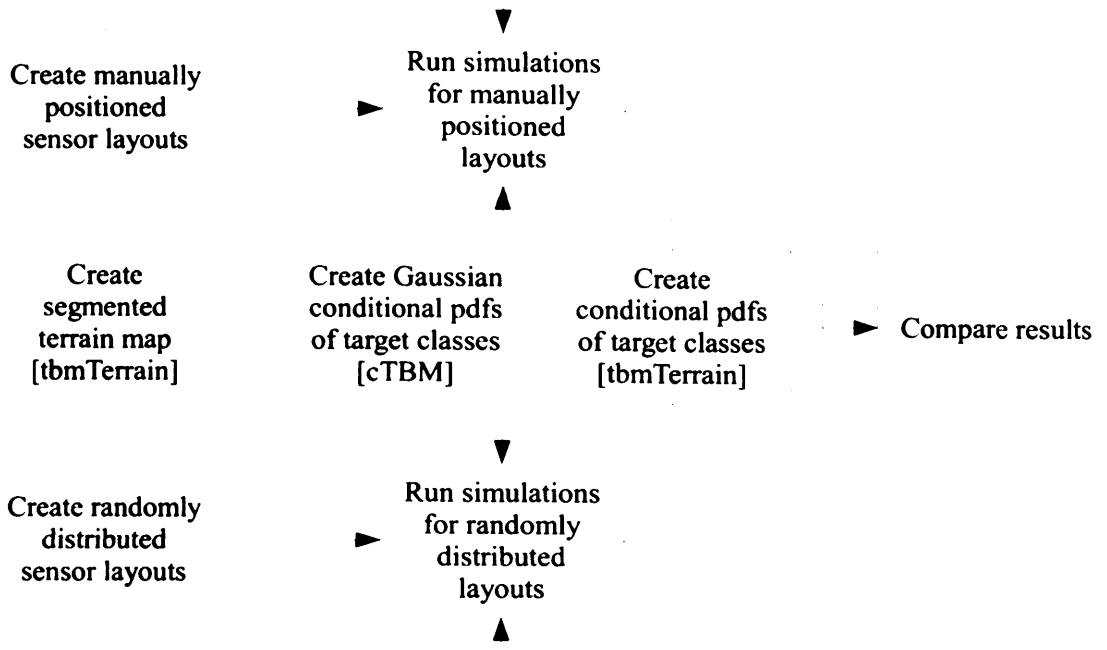


Figure 6.1: Methodology for comparing the performance of a tracking or JTAC algorithm when using different sensor positioning strategies. [cTBM] denotes the JTAC algorithm of Chapter 4, and [tbmTerrain] denotes the improved JTAC algorithm of Chapter 5.

optimise sensor deployment in an automated manner, but it could be extended to do this — this will be discussed in Section 6.4.

The method used for this investigation is described in Section 6.1. The results of experiments with two scenarios and the analysis of it can be found in Section 6.2. The conclusions this chapter are presented in Section 6.3. Section 6.4 provides possibilities for future work.

6.1 Wireless Sensor Network Deployment Method

Experiments will be carried out to prove the hypothesis that a WSN with sensors strategically positioned to take into account the terrain and therefore the likely target trajectories, will perform better than a WSN that uses randomly positioned sensors. Two scenarios will be used to test the hypothesis and demonstrate the use of our framework for WSN deployment planning; both of which use two sets of simulations — one set with sensors that have been deployed randomly, and another set with sensors that have been positioned manually (See Figure 6.1).

We will be comparing the results from the sets of simulations run with the two different sensor deployment strategies. The simulations that use manually positioned

sensors should show an increased tracking accuracy, and an improved classification accuracy as a result. Since the purpose of the WSN is to preform JTAC, an increase in tracking and classification accuracy for simulations that use strategically positioned sensors is clear indication of a better performance.

The set of simulations that use randomly positioned sensors are the simulations of Chapter 5. The set of simulations that use strategically positioned sensors will be run for this chapter, with sensor positions that have been determined experimentally. The two scenarios, Scenario A and Scenario B, are based on Scenarios A and B of Chapter 5 respectively.

For both scenarios, 15 manually positioned sensor layouts have been created and the results of using these layouts will be compared with their equivalent scenarios of Chapter 5 in which sensors were positioned using a uniform random distribution. For each scenario, the same target trajectory and simulation parameters as Chapter 5 will be used for a fair comparison. As with Chapter 5, horizon lengths of 5, 10, and 25 will be used. The WSN tracking approach of Williams et al. (2007), and the WSN JTAC approaches of Chapters 4 and 5 will be used for all horizon lengths.

In previous chapters, 100 sensor layouts were used for each combination of parameters; if a small number of layouts was used, a poorly distributed sensor layout could have a large effect on the overall results for a scenario. For the new simulations of this chapter, sensors are positioned manually, and as such it is not necessary to create such a large set of sensor layouts, and it would be time consuming to do so. Using 15 sensor layouts in each new simulation provides reliable results as long as the sensor layouts are of sufficient quality. As with Chapter 5, 20 sensors will be used in each simulation.

As with Scenario A of Chapter 5, Scenario A consists of an amphibious light tank travelling over road, grass, and water. The terrain and target trajectory for this scenario can be seen in Figure 5.4. The parameter values $R = 1.75$, $a = 1500000$, and $b = 200$ were used for all simulations. $\iota = 30$, $\mathcal{W} = 7$, and $\rho = 1.75$ were used with tbmTerrain simulations. Scenario A of Chapter 5 will be used for comparison. The modified version of Scenario A will *not* be used — and as such the bicycle target class is not included in the frame of discernment. Sensors are deployed in a manner that covers most of the region of interest, but does not cover the area of steep land and is biased towards a potential water crossing.

Scenario B consists of a car travelling along two sections of road with a roundabout connecting them. The terrain and target trajectory can be seen in Figure 5.21. All simulations used $R = 1.7$, $a = 1000000$, and $b = 200$. The tbmTerrain simulation parameters were $\iota = 20$, $\mathcal{W} = 7$, and $\rho = 1.15$. Sensors are positioned to track targets travelling on the road network, but are not limited to a specific route on the road network.

The JTAC approach of Chapter 4 was used in the same manner as in the previous chapter: a condition factor of $\gamma = 3.5$ was used, and a single *bba* is created at each time step. The *bba* is created using the speed extracted from μ_k.

In a similar way to previous chapters, for each combination of sensor positioning strategy, planning horizon length, scenario, and JTAC or tracking-only approach, a set of simulations will be run. For the strategically positioned sensor layouts this will consist of 15 simulations, and for the randomly positioned sensor layouts this will consist of 100 simulations — in both cases a different sensor layout will be used for each simulation within a set of simulations.

The results from a set of simulations will be combined — the distribution of these combined results will be used for comparison. The purpose of these experiments is to compare sets of simulations where the only difference is the sensor positioning strategy. A comparison will be made using the following metrics:

- Tracking accuracy
- Sensor utilisation
- Accrued communications costs
- Classification accuracy

Box plots will be used to provide a visual comparison of the distributions. It is expected that since the sets of simulations with strategically positioned sensor layouts are smaller, the variance of the results will be smaller than that of simulations that use randomly positioned sensor layouts. Variance will therefore not be used to evaluate the results.

As with previous chapters, tracking accuracy will be measured using the SMSE — the sum of the mean squared error at each time step between the estimated and ground truth target state. Sensor utilisation will be measured by taking the mean number of sensors that are active at a single time step. Classification accuracy will be measured by taking the mean of the classification probabilities of the ground truth target class for each time step. Accrued communications costs will simply be measured by adding the total communications cost for a simulation. Although a comparison of sensor utilisation and accrued communications costs will not be used to test the hypothesis, they will be performed to provide a greater insight into the differing behaviours that result from the two sensor positioning strategies. An example classification output from the two JTAC approaches using manually positioned sensor layouts will also be shown.

6.2 Results

This section presents a comparison of the results between simulations that used a random sensor deployment strategy and the simulations that used manually positioned sensors. The results for Scenarios A and B are shown, both of which show a different behaviour when using manually positioned sensors instead of randomly positioned sensors. The observed differences in WSN performance demonstrate the potential of using the framework presented in Chapter 3 for planning the deployment of sensors. The results for Scenario A can be found in Section 6.2.1, and the results for Scenario B are presented in Section 6.2.2.

An effect that was observed with simulations that used manually positioned sensors was that the processing time required to plan a typical time step appeared to be larger than that of simulations with randomly positioned sensors. This is because at each time step, a simulation with manually positioned sensors has more potential sensors to consider for use within the subset of active sensors as more sensors are generally closer to the object of interest.

6.2.1 Scenario A

Some of the figures provide a comparison between the simulations of Chapter 5 and the new simulations that have been run with the manually positioned sensor layouts — an '*' has been used to denote the latter in this case. The tracking performance of both sensor positioning strategies can be seen in Figure 6.2. A comparison of sensor usage can be found in Figure 6.3, and communications cost accrual in Figure 6.4. A comparison of classification performance can be found in Figure 6.5. The typical output of the TBM when using manually positioned sensors can be found in Figures 6.6 and 6.7 for the approaches of Chapters 5 and 4 respectively.

As with Chapters 4 and 5, the notation of the box plots is as follows: the box contains the interquartile range, the median is denoted by a red vertical line, the whiskers extend to the largest and smallest non-outlier data points, and a '+' is used to denote an outlier. A data point is considered an outlier if it is less than $q_1 - 1.5(q_3 - q_1)$ or more than $q_3 + 1.5(q_3 - q_1)$, where q_1 and q_3 are the first and third quartiles, respectively.

Figure 6.2 shows a clear improvement in tracking accuracy for manually positioned sensor layouts. This appears to be a result of increased sensor utilisation (See Figure 6.3) which could take place if more sensors can provide useful measurements whilst remaining within the communications cost constraint of the sensor selection algorithm. The increased sensor utilisation results in an increased in accrued communications cost (See Figure 6.4).

With $N = 5$ and 25, there is little difference between the classification accuracies for the positioning strategies of Chapter 5 and this chapter (See Figure 6.5). The results for $N = 10$ look quite different when comparing manually and randomly positioned sensors, but this is due to the differing sample sizes. It appears that there is a slight improvement in classification accuracy for tbmTerrain when using manually positioned sensors. The improvement in tracking performance with manually positioned sensor layouts results in a better assignment of belief within the TBM and hence an improved classification accuracy. With the JTAC approach of Chapter 4 there is little room for improvement in classification accuracy, but with tbmTerrain an improvement in classification accuracy is easier to obtain.

The typical output of the TBM for simulations with manually positioned sensor layouts cannot be directly compared with simulations with randomly positioned sensor layouts because the sensor layouts used in the comparison are not the same. However, it appears that the behaviour of the TBM with tbmTerrain is similar for manually and randomly positioned sensors, with perhaps a slightly better classification output with the use of manually positioned sensors. The TBM output for the approach of Chapter 4 appears to be quite different for the two sensor layout positioning strategies, but this is due to the different sensor positions. As with Chapter 5, both of the JTAC approaches show a period of confusion in classification. The tbmTerrain approach becomes confused as the target is slowing down on it's approach to the water. The approach of Chapter 4 becomes confused slightly later — this is due to the differing prior probabilities used in each approach.

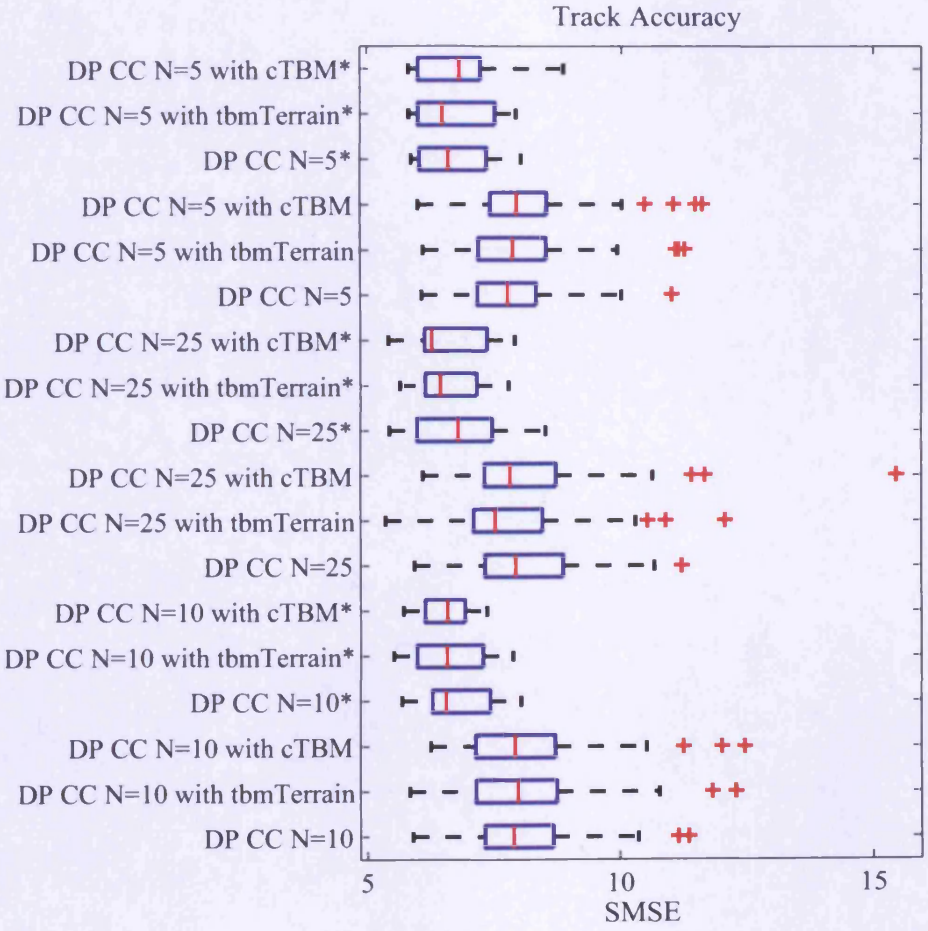


Figure 6.2: A comparison of track accuracies for Scenario A. Each box plot is created from a set of 100 Monte Carlo trials. A '*' denotes simulations with manually positioned sensor layouts, a lack of '*' denotes the use of randomly positioned sensors.

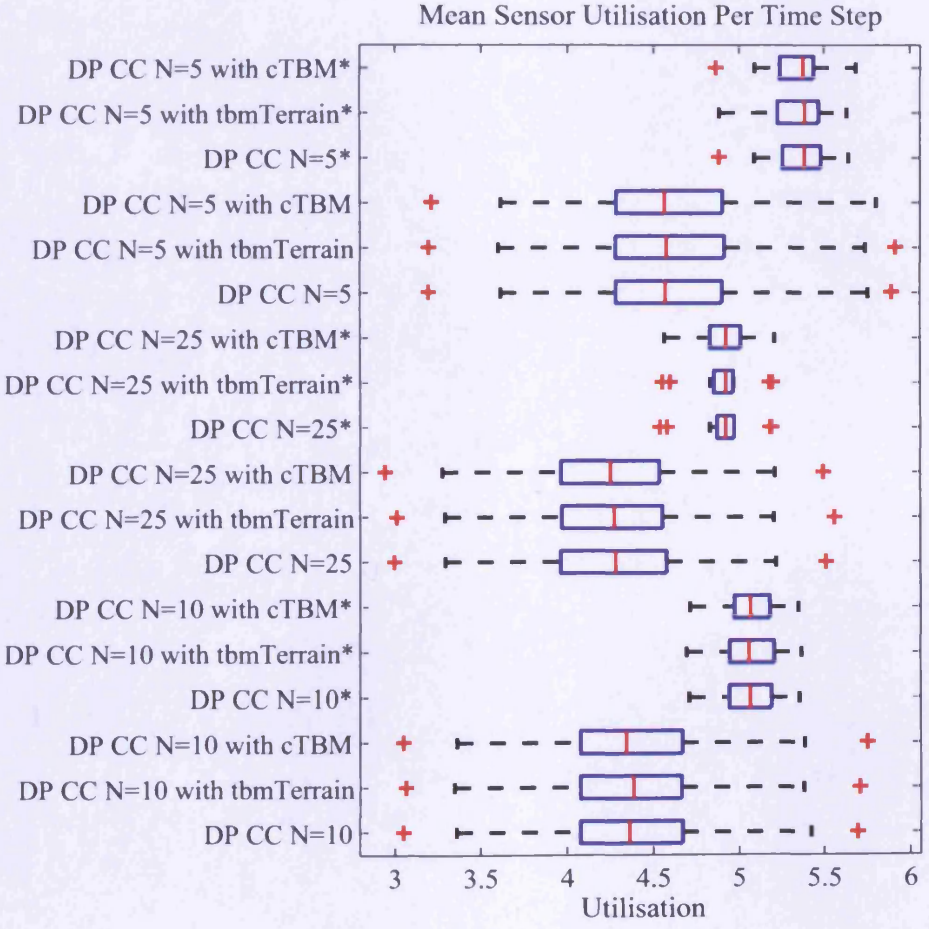


Figure 6.3: Mean number of sensors used per time step for Scenario A. Each box plot is created from a set of 100 Monte Carlo trials. A '*' denotes simulations with manually positioned sensor layouts, a lack of '*' denotes the use of randomly positioned sensors.

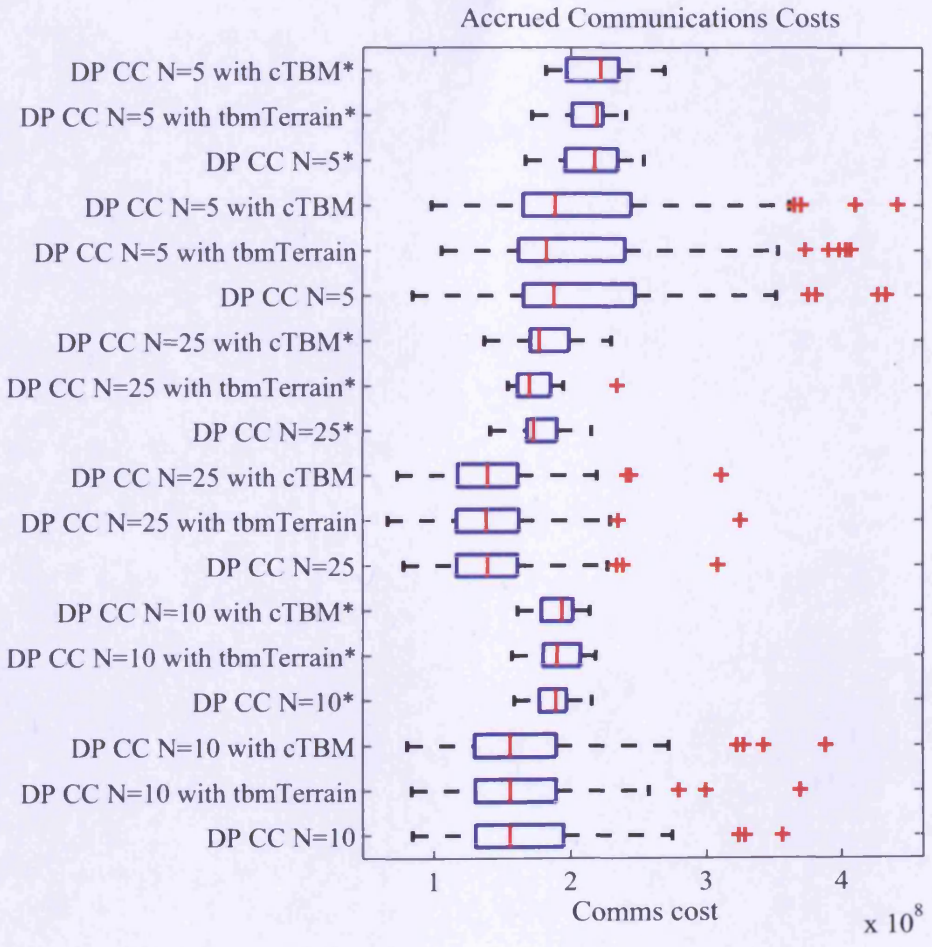


Figure 6.4: Accrued communications costs for Scenario A. Each box plot is created from a set of 100 Monte Carlo trials. A '*' denotes simulations with manually positioned sensor layouts, a lack of '*' denotes the use of randomly positioned sensors.

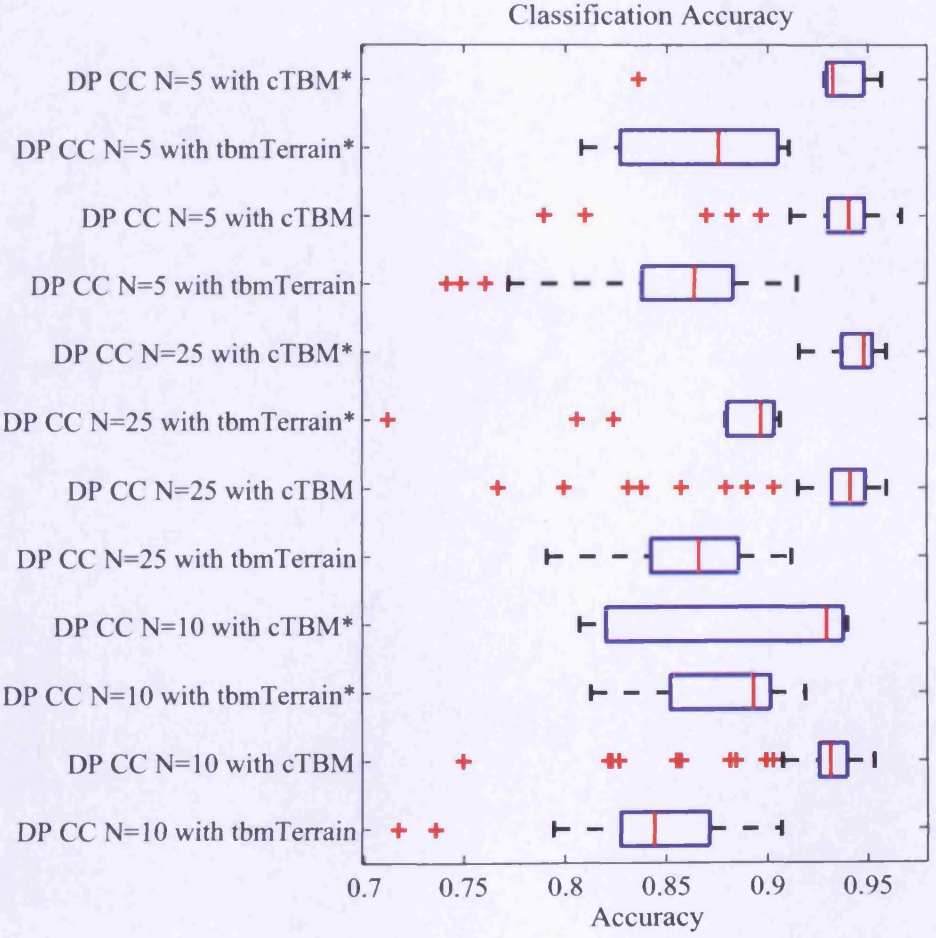


Figure 6.5: Classification accuracies for Scenario A. Each box plot is created from the mean value of $BetP(A)$ over time for each simulation. A '*' denotes simulations with manually positioned sensor layouts, a lack of '*' denotes the use of randomly positioned sensors.

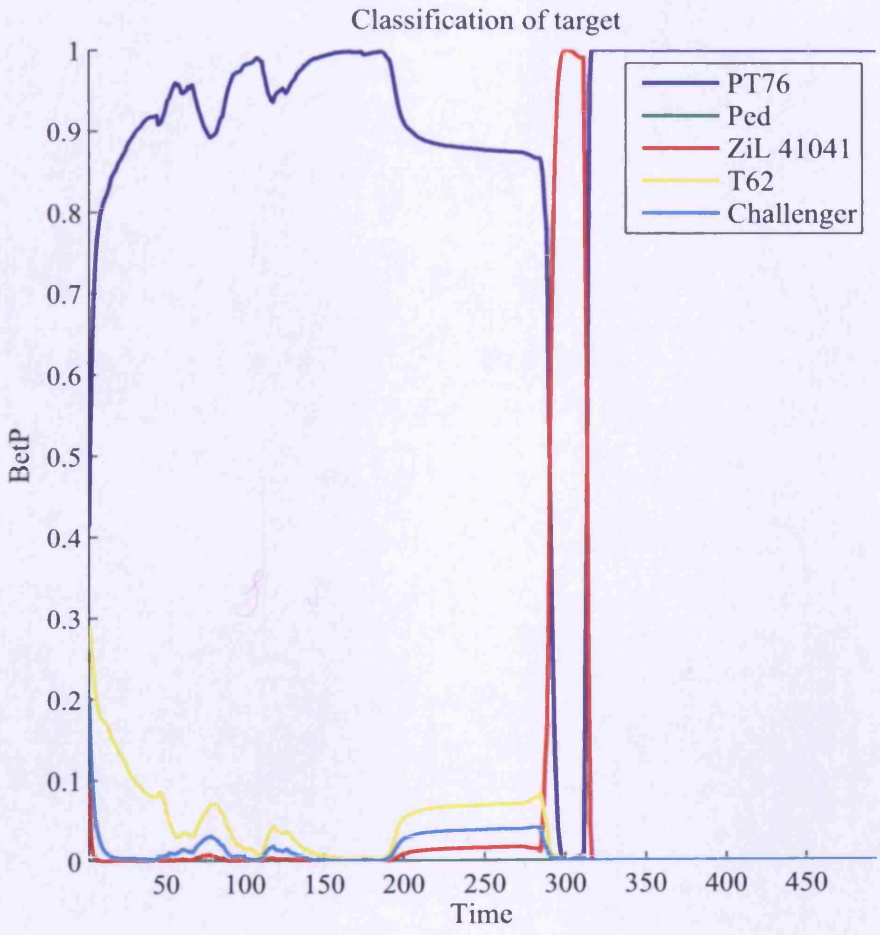


Figure 6.6: *BetP* for a typical Scenario A simulation using tbmTerrain with manually positioned sensor layouts.

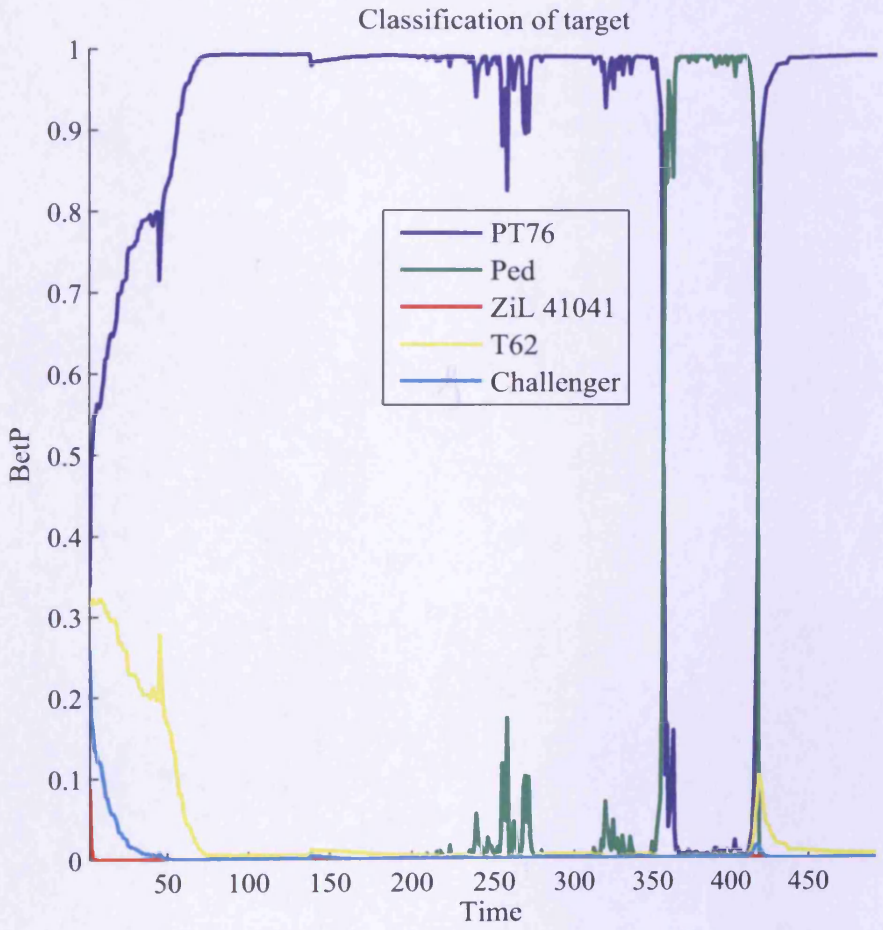


Figure 6.7: *BetP* for a typical Scenario A simulation using the approach of Chapter 4 with manually positioned sensor layouts.

6.2.2 Scenario B

As with Scenario A, some of the figures provide a performance comparison between the use of manually and randomly positioned sensors, and the remaining figures show a typical output of the TBM with both JTAC approaches. The tracking performance can be seen in Figure 6.8. Sensor utilisation can be seen in Figure 6.9 and the accrued communications cost in Figure 6.10. Figure 6.11 provides a comparison of classification performance. A typical output of the TBM can be seen in Figures 6.12 and 6.13 for the tbmTerrain approach and the approach of Chapter 4 respectively.

As with Scenario A, a clear improvement in tracking accuracy can be seen in Figure 6.8, which is due to the increased utilisation of sensors (See Figure 6.9) which are available near the target trajectory. The increased sensor usage has resulted in a larger accrued communications cost (See Figure 6.10).

There is no discernible difference in classification performance between sets of simulations that have used manually or randomly positioned sensors when comparing the same JTAC strategy (See Figure 6.11). Figures 5.29 and 6.12 indicates that an improvement in classification performance for a typical simulation using tbmTerrain is not possible; a maximum limit in classification accuracy is already quickly achieved by simulations in Scenario B of Chapter 5. It is likely that this limit has been achieved because of a sufficiently good tracking accuracy. Similarly, Figure 6.11 indicates a similar classification accuracy for simulations using randomly or manually positioned sensors — indicating that in improvement in tracking accuracy is not enough to improve classification accuracy.

A comparison between Figures 5.30 and 6.13 shows similar behaviour for the TBM when using the JTAC approach of Chapter 4 regardless of whether sensors are positioned randomly or manually. It appears the simulations that run with randomly positioned sensors already produce an accurate enough estimate of the kinematic state of the target; any improvement in tracking accuracy will not have a big impact on the assignment of belief within the TBM. The poor performance of the JTAC approach is due to the selection of the prior probabilities.

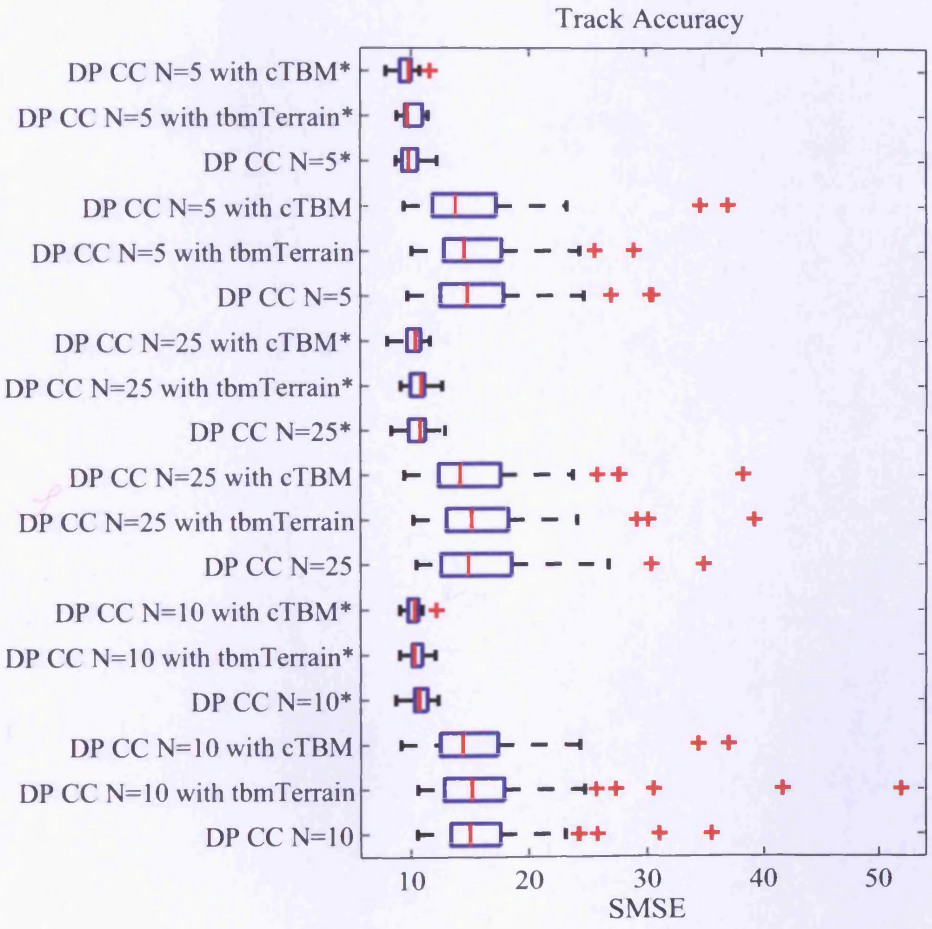


Figure 6.8: A comparison of track accuracies for Scenario B. Each box plot is created from a set of 100 Monte Carlo trials. A '*' denotes simulations with manually positioned sensor layouts, a lack of '*' denotes the use of randomly positioned sensors.

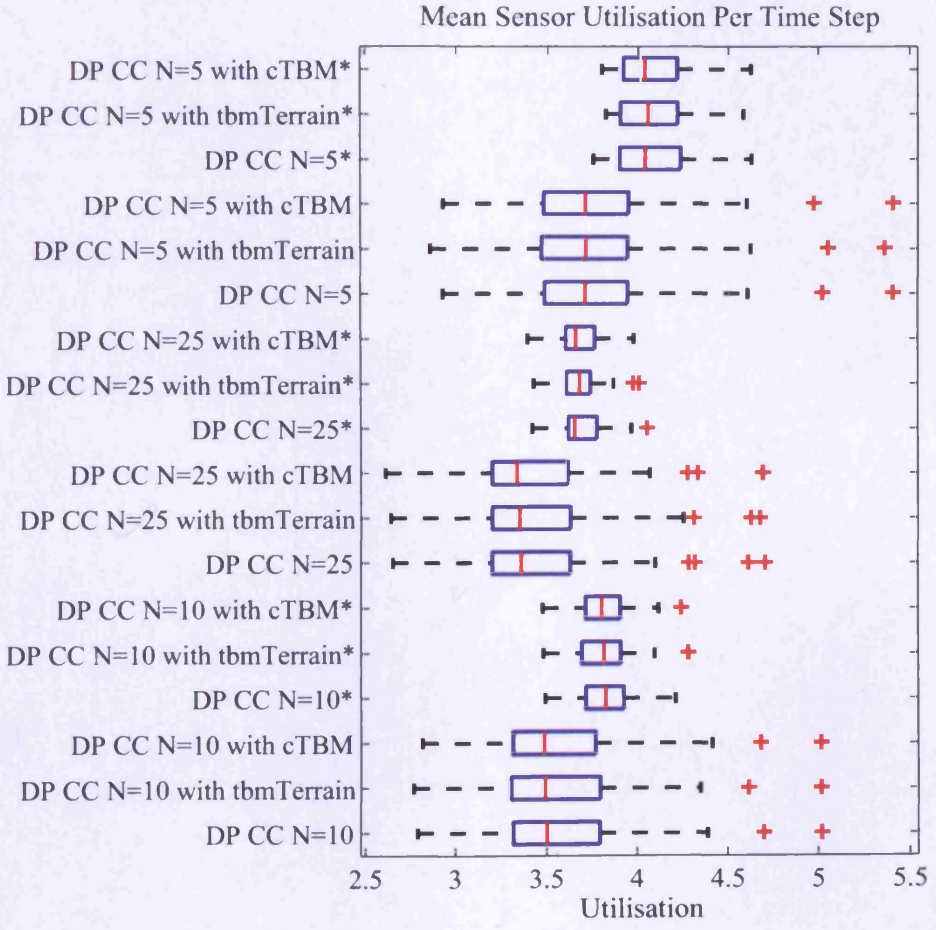


Figure 6.9: Mean number of sensors used per time step for Scenario B. Each box plot is created from a set of 100 Monte Carlo trials. A '*' denotes simulations with manually positioned sensor layouts, a lack of '*' denotes the use of randomly positioned sensors.

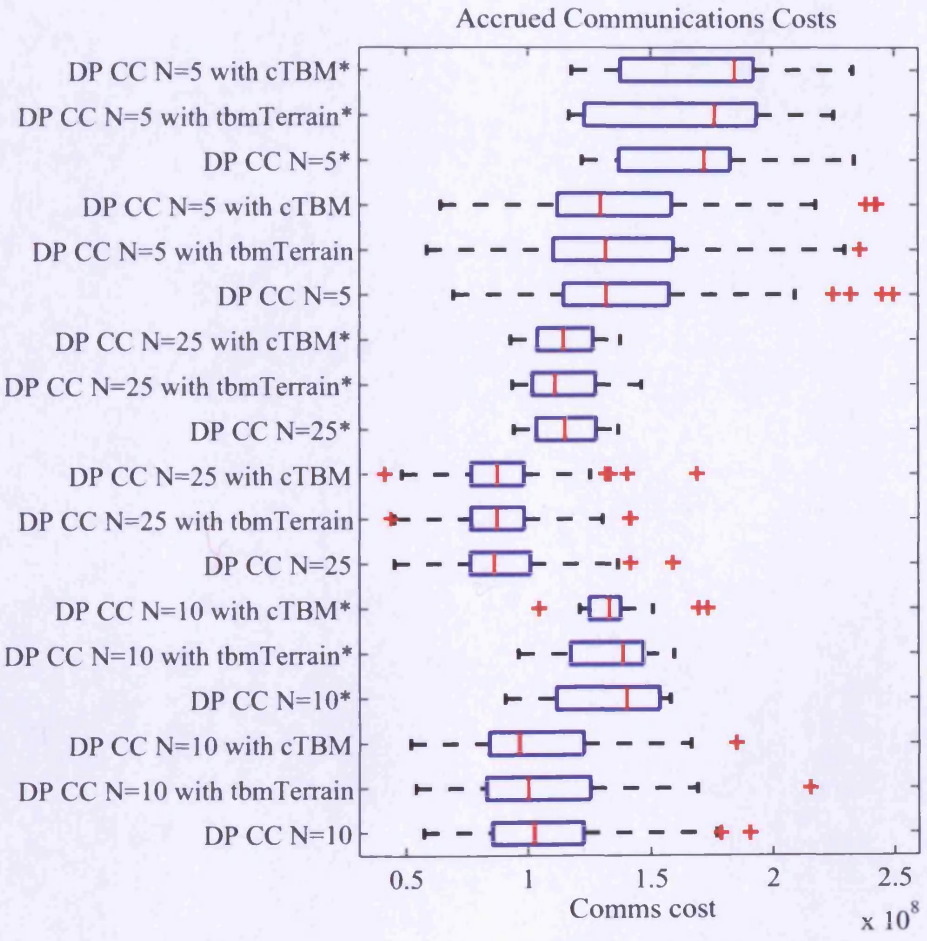


Figure 6.10: Accrued communications costs for Scenario B. Each box plot is created from a set of 100 Monte Carlo trials. A '*' denotes simulations with manually positioned sensor layouts, a lack of '*' denotes the use of randomly positioned sensors.

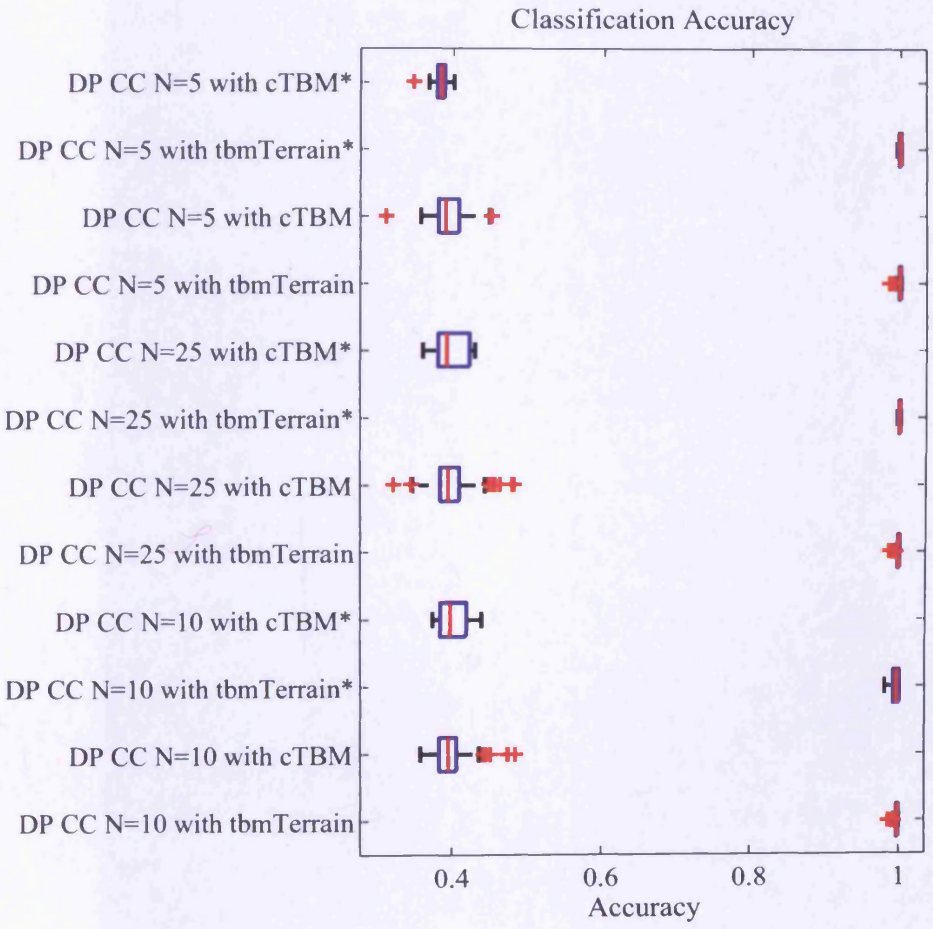


Figure 6.11: Classification accuracies for Scenario B. Each box plot is created from the mean value of $BetP(A)$ over time for each simulation. A '*' denotes simulations with manually positioned sensor layouts, a lack of '*' denotes the use of randomly positioned sensors.

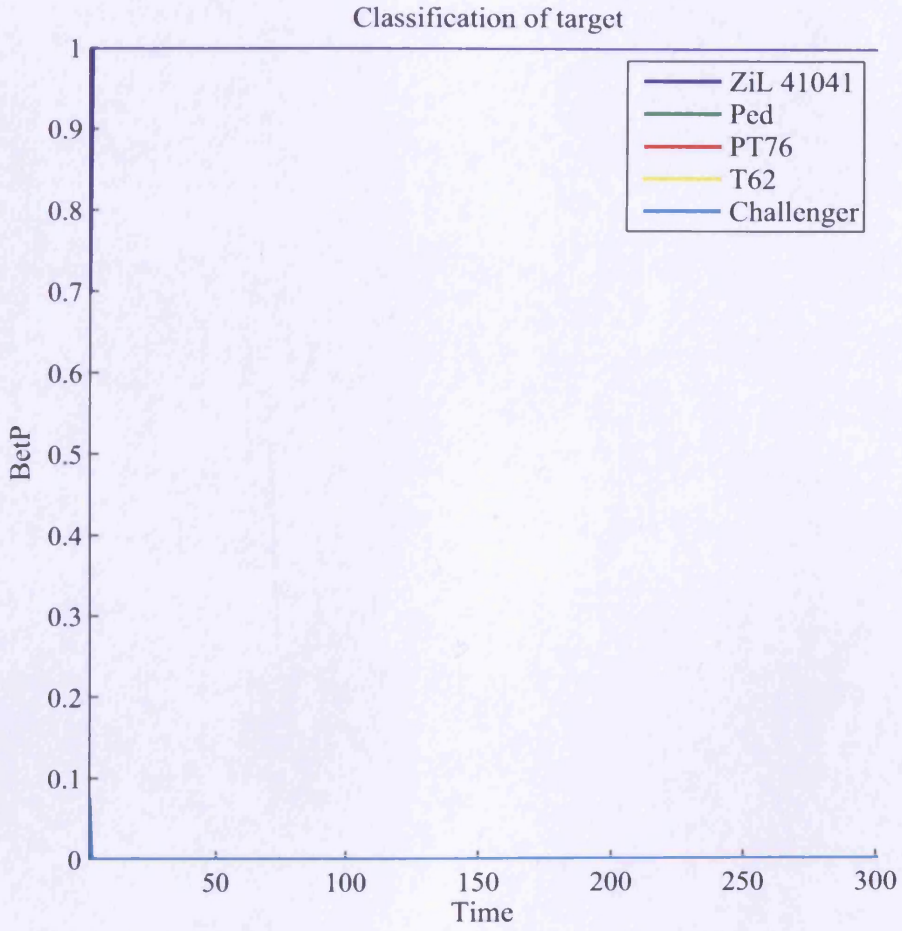


Figure 6.12: *BetP* for a typical Scenario B simulation using tbmTerrain with manually positioned sensor layouts.

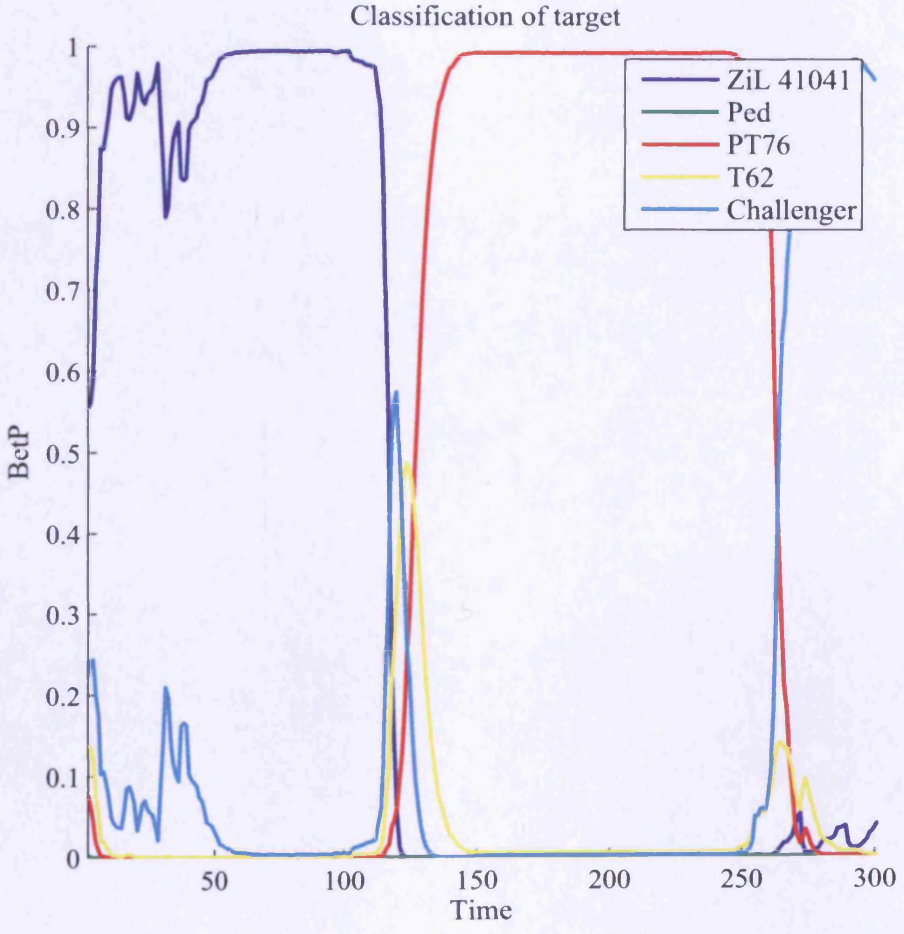


Figure 6.13: *BetP* for a typical Scenario B simulation using the approach of Chapter 4 with manually positioned sensor layouts.

6.3 Conclusions

This chapter has demonstrated the use of the framework presented in Chapter 3 to aid sensor deployment planning. We hypothesised that the use of manually positioned sensors would result in a better tracking performance and hence a better classification accuracy. The results have shown that tracking accuracy has been improved by manually positioning sensors, but this has not necessarily been the case for classification accuracy. An increase in sensor usage, and hence communications cost accrual has also been observed. In our simulations, the typical output of the TBM for both JTAC strategies has not changed significantly with the use of manually positioned sensors.

It is obvious that an increase in tracking performance will be seen when using manually positioned sensors. Sensors are positioned to take into account the likely target trajectories which can be estimated from the terrain; more sensors are close to the target, providing useful measurements at a relatively low cost. Evidence of this can be seen in the results, where the number of sensors used at each step is usually higher for simulations with manually positioned sensors.

A less expected effect of using manually positioned sensors is that an increase in communications cost is observed; at a typical time step, more sensors are close to the expected location of the target — each of which has a greater potential to provide a reduction in uncertainty whilst remaining within the communication cost constraints when compared to randomly positioned sensors that will probably be further away. If information constrained sensor selection was used instead of communication constrained then a reduction in communication costs should be seen, but possibly with a reduction in tracking accuracy. The use of our framework for sensor deployment planning provides the possibility of avoiding these unexpected costs in the real world by observing them in simulations.

It was expected that an improvement in classification accuracy would occur when using manually positioned sensors — this has not generally been the case. The likely reason for this is that the estimates of the kinematic state of the target from the particle filter were already of a sufficiently good accuracy; any improvement in this accuracy has not had a large impact in the assignment of belief within the TBM.

6.4 Future Work

There are many possibilities for future investigations to extend this work, including:

- The use of heterogeneous sensors

- Implementing information constrained sensor management
- Using a lower SNR ratio in a repeated batch of simulations
- An improvement to the current framework to model battery usage and sensor failure
- The pruning of sensor layouts to reduce costs
- Automating sensor deployment planning

6.4.1 Heterogeneous Sensors

All of the experiments carried out for this thesis have used homogeneous sensors — every sensor in a single simulation has used the same sensor model and parameters, and the same communications parameters. Further experiments could use this framework to experiment with the trade-off between the use of expensive, high-quality sensors, and cheap, lower quality sensors. Like the simulations of Chapters 4 and 5, lower cost sensors could be positioned randomly — as if they are disposable sensors that have been deployed from an aircraft. The more expensive sensors could be manually positioned — as if they are sensors of high value that have been deployed by personnel. It would be interesting to compare the tracking and classification performance of networks of the two different types of sensors, and perhaps mixtures of the two.

6.4.2 Information Constrained Sensor Management

As stated in Section 6.3, it is expected that the use of information constrained sensor management would result in a different sensor usage to that of communication constrained management. Further experiments could investigate this hypothesis — providing a valuable insight into the effect that the sensor selection algorithm can have on the performance of simulations within this JTAC framework.

6.4.3 Using a Lower Signal to Noise Ratio

Experiments have shown an increase in tracking performance but no clear improvement in classification accuracy. As previously hypothesised, this may be due to the kinematic state estimates of the target already being of a sufficiently good quality. It would be interesting to confirm this by re-running the experiments of this chapter (and Chapter 5) with a lower SNR. The results from the simulations would hopefully show a

better tracking and classification accuracy for simulations that use manually positioned sensors.

6.4.4 Modelling Battery Usage and Sensor Failure

Throughout this thesis, simulations have assumed that all sensors have an unlimited amount of energy and never fail — this is due to the use of the sensor management algorithm by Williams et al. (2007). An improvement to this algorithm could be inclusion of a limited energy budget for each sensor and the random loss of sensors — as if nodes were battery powered and subject to environmental influences. This would be a non-trivial task to accomplish as planning takes place from the end of the planning horizon back to the current time step, and a sensor may have enough power to transmit a observation a short distance, but not a longer one. The algorithm by Williams et al. is also not designed to cope with the loss of a node, especially whilst it is a leader node — modifications would be required to this algorithm if the framework was improved to allow the possibility of any node to fail at any time step. One potential solution could be the use of a backup leader node. The backup leader node, which could be the second most suitable leader node, could take over from the leader node if a number of regular messages are not received. These messages could be observations from active sensors (via the leader node) or probabilistic state updates — providing not only a means to signal a functioning leader node, but also a means of keeping the backup leader node up to date.

6.4.5 Pruning Unused Sensors

The work of this chapter could be extended to prune sensors from layouts to reduce deployment costs. Simulations of different but likely target trajectories with the same sensor layout could highlight sensors that have little or no utilisation. The removal of these poorly positioned sensors, and the subsequent reduction in the number of sensors would lower the computational effort required to plan sensor usage in real world deployment.

6.4.6 Automating Sensor Deployment Planning

This chapter has demonstrated the use of the framework presented in Chapter 3 to aid sensor deployment planning. It would be possible to extend this work to automate sensor deployment planning. This could be seen as an optimisation problem — the sensor positions would need to be optimised for a given set of likely target trajectories, a

fixed number of sensors, and a specified set of simulation parameters. The optimisation process would seek to maximise a metric that combines both tracking and classification accuracy. This task would not be trivial due to the complex nature of the optimisation surface.

A simpler, alternative approach could be to construct a probability distribution from which sensor positions could be drawn. The distribution would reflect the likelihood that a sensor is utilised in a typical simulation. A number of simulations with randomly positioned sensors could be run in parallel, using a system such as Condor. The sensor utilisation statistics could then be used to build the probability distribution. Regions of high sensor utilisation would result in local maxima in the distribution — increasing the likelihood in a sensor being positioned in the region.

Chapter 7

Conclusions and Future Work

This chapter summarises the research presented in this thesis and has proposes future work to extend this. The research of this thesis has focused on JTAC with WSNs. A framework has been created to perform tracking or JTAC with WSN whilst managing node usage to balance the quality of information with the cost of obtaining it. It is essential to manage energy usage within a WSN in order to maximise the lifetime of the network.

The work presented is based on a combination of a WSN management algorithm by Williams et al. (2007), and a joint tracking and classification algorithm by Powell et al. (2006). This integration has been extended to perform JTAC with a WSN in more realistic scenarios. All of the scenarios have been land-based with a single target in each one. A particle filter has been used for tracking, and the TBM has been used for classification.

The thesis has been organised as follows:

- Chapter 2 provided an overview of relevant topics. Topics discussed include tracking, classification and sensor management — all of which are essential to the research presented in the later chapters of this thesis.

- Chapter 3 presented the framework in which the algorithms were implemented. Further chapters utilised the framework to compare the performance of algorithms.

- Chapter 4 presented a novel method to jointly track and classify a target with a WSN, whilst also planning the usage of sensors in a such a way that balances the cost of communications with the quality of information obtained by the sensors.

- Chapter 5 improved upon Chapter 4 to perform JTAC in more realistic scenarios. Novel improvements included the use of terrain information, the use of an ellipti-

cal area of a terrain map to account for position uncertainty and its subsequent use within the classification stage, and the addition of an 'intelligent memory' within the classification. The results have shown a consistently good classification performance with these improvements — this was not the case without them.

- Chapter 6 has shown how the framework presented in Chapter 3 can be used as a tool to aid sensor deployment planning. This was demonstrated with a comparison of WSN Joint Tracking and Classification performance using two different sensor deployment strategies — where the results showed the tracking performance of a WSN to be better when sensors where strategically positioned close to the expected path of the target.

In summary, the main contributions of this thesis have been:

- A novel framework for performing tracking or joint tracking and classification with a WSN — and its demonstrated use in both developing new algorithms and as a tool to aid sensor deployment.
- A novel method for jointly tracking and classifying a target with a WSN whilst managing the usage of sensors.
- Novel improvements to the above method to perform joint tracking and classification in more realistic scenarios by:
  - Considering, within the classification process, the uncertainty of the target's kinematic state, the terrain, and how terrain restricts a target's movement.
  - Adding an 'intelligent memory" to the TBM to improve the output of classifications that are updated iteratively over time; preventing the assignment of belief to target classes that past behaviour has shown are not feasible.

In conclusion, this thesis has proposed novel contributions to perform JTAC in a new framework, has improved the framework for more realistic scenarios, and demonstrated its use as a tool to aid sensor deployment. We have shown that our framework has the flexibility to be used in not only the development and testing of algorithms, but also as an aid for sensor deployment planning. The modular framework has been used with both simple scenarios, and more advanced scenarios that contain terrain and non-linear target motion. The next section discusses potential improvements to this work.

7.1 Future Work

Various improvements have been discussed in Chapters 4–6, some of which have already been addressed, and some haven't. A number of the latter improvements are discussed in this section, this includes improving the feedback from the particle filter to the TBM; investigating the effect of a compromise used within the classification stage; extending work to perform multi-target tracking; the use of open world classification scenarios; improving the realsim of simulations; and automating sensor deployment planning.

The two methods presented in Chapters 4 and 5 for providing feedback from the TBM to the particle filter unfortunately did not provide the desired improvement in tracking accuracy. A better method is required to perform this feedback. Ideally, the method should not introduce bias and should not assume the state estimate can be adequately modelled with a Gaussian pdf. The former occurred with the feedback method used in Chapter 4, and the latter in Chapter 5. Creating such a feedback method will not be a trivial task, but could improve tracking accuracy by updating the state estimate to reflect what the target class is capable of.

An alternative feedback method could be to change the particle filter tuning parameters to reflect the target class. For example, although all of the target classes used the same state update equation, tracking may have been improved by using a different magnitude of process noise for each class to reflect its manoeuvrability.

In Chapter 5, a modification was made to create a single *bbm* from the particle filter estimate at each time step rather than a *bbm* for each particle. This compromise between richness and computational complexity was necessary due long the execution times of simulations. It would be interesting to investigate the difference in classification output between the two methods — especially with the use of different combination rules and a poorer observation SNR.

This thesis has focused on single target tracking, but a more realistic system may require the ability to track multiple targets at any one time. Modifying the existing algorithms to track multiple targets would not be a trivial task; two of the most challenging problems would be the modifications required to perform data association and plan WSN node usage. This would be particularly important when two targets are close together — especially if they have similar kinematic states, such as targets travelling in formation.

All of the classification performed in experiments have assumed a closed world. This is largely due to the iterative nature of the classification system. A system designed for real world applications could not use such an assumption, but in doing so would need

to cope with the assignment of belief to the empty set, $\emptyset$. This assignment would occur when there is disagreement between *bbas* — which always occurs due to the use of noisy observations.

Simulations were the most suitable method of analysing the performance of algorithms developed in this thesis — the arguments for doing this have been discussed in Section 3.2. Some aspects of WSN were not modelled in the simulations or algorithms, including limited battery life and WSN node failure — in order to design and test algorithms for use in more realistic scenarios, these aspects should be modelled. Modifying the existing code to limit a node's battery life and to produce random or deterministic node failure would be a relatively easy task, but modifying the WSN management algorithm, by Williams et al. (2007), would not be. This is because sensor selection takes place along a planning horizon, from the end of the planning horizon back toward the current time step — this problem was discussed in more detail in Section 6.4.4.

Chapter 6 demonstrated the use of the joint tracking and classification framework used throughout this thesis to aid sensor deployment planning. As discussed in Section 6.4.6, further work could be carried out to extend this work to perform automated sensor deployment planning. The use of simulations to aid or automate sensor deployment planning could be a very attractive option for optimising sensor deployment strategies — especially compared to using real world experiments to determine good strategies. An automated sensor deployment planning system would be of most use if it adequately modelled the environment in which the real WSN would be deployed. In order to do this, many of the improvements proposed in this section, such as multi-target tracking and modelling node failure, would be required.

Bibliography

Akyildiz, I. F., Su, W., Sankarasubramaniam, Y., and Cayirci, E. (2002). Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422.

ALERT Systems (2005). ALERT Systems homepage. `http://www.alertsystems.org/` [Accessed 19 Oct 2010].

Arora, A., Dutta, P., Bapat, S., Kulathumani, V., Zhang, H., Naik, V., Mittal, V., Cao, H., Demirbas, M., Gouda, M., Choi, Y., Herman, T., Kulkarni, S., Arumugam, U., Nesterenko, M., Vora, A., and Miyashita, M. (2004). A line in the sand: a wireless sensor network for target detection, classification, and tracking. *Computer Networks*, 46(5):605–634.

Arulampalam, S., Maskell, S., Gordon, N., and Clapp, T. (2002). A tutorial on particle filters for on-line non-linear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188.

Atmel Corporation (2010). ATmega128/L datasheet. `http://www.atmel.com/dyn/resources/prod_documents/doc2467.pdf` [Accessed 23 Oct 2010].

Bauer, M. (1997). Approximation algorithms and decision making in the dempster-shafer theory of evidence – an empirical study. *International Journal of Approximate Reasoning*, 17(2-3):217 – 237.

Bayes, T. (1763). An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London*, (53):370–418.

Brooks, R. R., Ramanathan, P., and Sayeed, A. (2003). Distributed target classification and tracking in sensor networks. *Proceedings of The IEEE*, 91(8):1163–1171.

Caron, F., Ristic, B., Duflos, E., and Vanheeghe, P. (2006). Least committed basic belief density induced by a multivariate gaussian pdf. In *Proceedings of International Conference on Information Fusion 2006*.

Caron, F., Smets, P., Duflos, E., and Vanheeghe, P. (2005). Multisensor data fusion in the frame of the TBM on reals. application to land vehicle positioning. In *Proceedings of International Conference on Information Fusion 2005*, volume 2.

Challa, S. and Pulford, G. W. (2001). Joint target tracking and classification using radar and ESM sensors. *IEEE Transactions on Aerospace and Electronic Systems*, 37(3):1039–1055.

Chu, M., Haussecker, H., and Zhao, F. (2002). Scalable information-driven sensor querying and routing for ad hoc heterogeneous sensor networks. *International Journal of High Performance Computing Applications*, 16(3):293–313.

Condor Team (2009a). *Condor Version 7.2.5 Manual*. University of Wisconsin-Madison. `http://www.cs.wisc.edu/condor/manual/v7.2/condor-V7_2_5-Manual.pdf` [Accessed 2 Aug 2010].

Condor Team (2009b). Directed acyclic graph manager. `http://www.cs.wisc.edu/condor/dagman/` [Accessed 13 Aug 2010].

Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory*. John Wiley & Sons.

Crossbow Technology Inc. (2005). Cricket series mote. `http://www.xbow.com/Products/productsdetails.aspx?sid=116` [Accessed 3 Feb 2006].

Crossbow Technology Inc. (2006a). MICA2 wireless measurement system. `http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf` [Accessed 27 Apr 2006].

Crossbow Technology Inc. (2006b). *MPR/MIB Users Manual*. Crossbow Technology Inc.

Crossbow Technology Inc. (2006c). Telosb. `http://www.willow.co.uk/TelosB_Datasheet.pdf` [Accessed 19 Oct 2010].

Crossbow Technology Inc. (2007). MICA2DOT wireless microsensor mote. URL:*http://www.xbow.com/Products/Product_pdf_files/Wireless_pdf/-MICA2DOT_Datasheet.pdf* [Accessed 8 Jan 2007].

Delmotte, F. and Smets, P. (2004). Target identification based on the transferable belief model interpretation of dempster-shafer model. *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, 34(4):457–471.

Dempster, A. P. (1968). A generalization of bayesian inference. *Journal of the Royal Statistical Society*, 30(2):205–247.

Dezert, J. and Smarandache, F. (2009). An introduction to DSmT. *CoRR*, abs/0903.0279.

Doré, P. E. and Martin, A. (2010). About using beliefs induced by probabilities. In *Workshop on the theory of belief functions*, Brest, France.

Douc, R., Cappé, O., and Moulines, E. (2005). Comparison of resampling schemes for particle filtering. In *Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis*, pages 64–69. Citeseer.

Doucet, A., de Freitas, N., and Gordon, N. (2001a). *An Introduction to Sequential Monte Carlo Methods*, chapter 1. In (Doucet et al., 2001b).

Doucet, A., de Freitas, N., and Gordon, N., editors (2001b). *Sequential Monte Carlo Methods in Practice*. Springer, New York, NY, USA.

Dubois, D. and Prade, H. (1988). Representation and combination of uncertainty with belief functions and possibility measures. *Computational Intelligence*, 4(3):244–264.

Ertin, E., Fisher, J., and Potter, L. (2003). Maximum mutual information principle for dynamic sensor query problems. In *Information Processing in Sensor Networks*, volume 3 of *Lecture Notes in Computer Science*, pages 405–416. Springer-Verlag.

Fiche, A., Martin, A., Cexus, J.-C., and Khenchaf, A. (2010). Continuous belief functions and α-stable distributions. In *Proceedings of International Conference on Information Fusion 2010*, Edinburgh, UK.

Fosbury, A. M., Singh, T., Crassidis, J. L., and Springen, C. (2007). Ground target tracking using terrain information. In *Proceedings of International Conference on Information Fusion 2007*, pages 1–8.

García-hernández, C. F., Ibargüengoytia-gonzález, P. H., García-hernández, J., and Pérez-díaz, J. A. (2007). Wireless sensor networks and applications: a survey. *International Journal of Computer Science and Network Security*, 7(3):264–273.

Gordon, N. J., Maskell, S., and Kirubarajan, T. (2002). Efficient particle filters for joint tracking and classification. In *Proceedings of SPIE*, volume 4728, pages 439–449. SPIE.

Gordon, N. J., Salmond, D. J., and Smith, A. F. M. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. In *Radar and Signal Processing, IEE Proceedings F*, volume 140, pages 107–113. IEE.

Haenggi, M. (2005). Opportunities and challenges in wireless sensor networks. In Ilyas, M. and Mahgoub, I., editors, *Handbook of Sensor Networks: Compact Wireless and Wired Systems*, chapter 1. CRC Press.

Hussain, S., Schaffner, S., and Moseychuck, D. (2009). Applications of wireless sensor networks and rfid in a smart home environment. In *Proceedings of of the Annual Conference Communication Networks and Services Research*, pages 153–157, Los Alamitos, CA, USA. IEEE Computer Society.

Jamieson, A., Breslin, S., Nixon, P., and Smeed, D. (2004). M_iPOS - the mote indoor positioning system. In *1st International Workshop on Wearable and Implantable Body Sensor Networks*.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45.

Khan, J. M., Katz, R. H., and Pister, K. S. J. (2000). Emerging challenges: Mobile networking for "smart dust". *Journal of Communications and Networks*, 2(3):188–196.

Kong, K., Liu, J. S., and Wong, W. H. (1994). Sequential imputations and bayesian missing data problems. *Journal of the American Statistical Association*, 89(425):278–288.

Li, X. R. and Jilkov, V. P. (2003). Survey of maneuvering target tracking. part I: Dynamic models. *IEEE Transactions on Aerospace and Electronic Systems*, 39(4):1333–1364.

Liu, J. S., Chen, R., and Logvinenko, T. (2001). *A Theoretical Framework for Sequential Importance Sampling with Resampling*, chapter 11. In (Doucet et al., 2001b).

Lowrance, J. D., Garvey, T. D., and Strat, T. M. (1986). A framework for evidential-reasoning systems. In *Proceedings of AAAI-86*, pages 896–903.

Maroti, M., Simon, G., Ledeczi, A., and Sztipanovits, J. (2004). Shooter localization in urban terrain. *Computer*, 37(8):60 – 61.

Martin, A. (2009). Implementing general belief function framework with a practical codification for low complexity. In Smarandache, F. and Dezert, J., editors, *Advances and Applications of DSmT for Information Fusion*, volume 3, chapter 7, pages 217–274. American Research Press.

Martin, A. and Osswald, C. (2006). A new generalization of the proportional conflict redistribution rule stable in terms of desicion. In (Smarandache and Dezert, 2006a), chapter 2, pages 69–88.

Martin, A. and Osswald, C. (2007). Toward a combination rule to deal with partial conflict and specificity in belief functions theory. In *Proceedings of International Conference on Information Fusion 2007*, Québec, Canada.

Maskell, S. (2008). A bayesian approach to fusing uncertain, imprecise and conflicting information. *Information Fusion*, 9(2):259–277.

Maybeck, P. S. (1979). *Stochastic models, estimation, and control*, volume 1 of *Mathematics in Science and Engineering*. Academic Press.

Mercier, D., Quost, B., and Denœux, T. (2005). Contextual discounting of belief functions. In Godo, L., editor, *Symbolic and Quantitative Approaches to Reasoning with Uncertainty*, volume 3571 of *Lecture Notes in Computer Science*, pages 472–472. Springer Berlin / Heidelberg.

Ordnance Survey (2005). Carmarthen & Kidwelly. [map] 1:25,000, 4 cm to 1 km. Ordanance Survey, Southampton, UK.

Ordnance Survey (2007). Brecon Beacons national park, western area. [map] 1:25,000, 4 cm to 1 km. Ordanance Survey, Southampton, UK.

Osswald, C. and Martin, A. (2006). Understanding the large family of dempster-shafer theory's fusion operators - a decision-based measure. In *Proceedings of International Conference on Information Fusion 2006.*

PermaSense Project (2010). PermaSense homepage. `http://www.permasense.ch/` [Accessed 14 Nov 2010].

Pottie, G. J. and Kaiser, W. J. (2000). Wireless integrated network sensors. *Communications of the ACM*, 43(5):51–58.

Powell, G. and Marshall, D. (2005). Joint tracking and classification of nonlinear trajectories of multiple objects using the transferable belief model and multi-sensor fusion framework. In *Proceedings of International Conference on Information Fusion 2005*, volume 2.

Powell, G., Marshall, D., Smets, P., Ristic, B., and Maskell, S. (2006). Joint tracking and classification of airbourne objects using particle filters and the continuous transferable belief model. In *Proceedings of International Conference on Information Fusion 2006*.

Powell, G., Roberts, M., and Marshall, D. (2010a). Empty set biasing issues in the Transferable Belief Model for fusing and decision making. In *Proceedings of International Conference on Information Fusion 2010*, Edinburgh, UK.

Powell, G., Roberts, M., and Marshall, D. (2010b). Pitfalls for recursive iteration in set based fusion. In *Workshop on the Theory of Belief Functions*, Brest, France.

Powell, G., Roberts, M., and Owen, T. (2010c). Transferable belief models for human welfare. In *Proceedings of International Conference on Information Fusion 2010*, Edinburgh, UK.

Qi, H., Wang, X., Iyengar, S., and Chakrabarty, K. (2001). Multisensor data fusion in distributed sensor networks using mobile agents. In *Proceedings of International Conference on Information Fusion 2001*, pages 11–16.

Ristic, B., Arulampalam, S., and Gordon, N. (2004a). *Beyond the Kalman Filter*. Artech House.

Ristic, B., Arulampalam, S., and Gordon, N. (2004b). *A Tutorial on Particle Filters*, chapter 3. In (Ristic et al., 2004a).

Ristic, B. and Smets, P. (2004). Belief function theory on the continuous space with an application to model based classification. *Proceedings of Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU*, pages 4–9.

Robert, C. P. (2007). *The Bayesian Choice*. Springer, New York, NY, USA, 2nd edition.

Roberts, M. and Marshall, D. (2008). Using classification to improve wireless sensor network management with the continuous transferable belief model. In Carapezza, E. M., editor, *Unmanned/Unattended Sensors and Sensor Networks V*, volume 7112, page 711204, Cardiff, UK. SPIE.

Roberts, M., Marshall, D., and Powell, G. (2010). Improving joint tracking and classification with the Transferable Belief Model and terrain information. In *Proceedings of International Conference on Information Fusion 2010*, Edinburgh, UK.

Sallai, J., Balogh, G., Maróti, M., Lédeczi, Á., and Kusy, B. (2004). Acoustic ranging in resource-constrained sensor networks. In *Proceedings of International Conference on Wireless and Mobile Computing*.

Shafer, G. (1976). *A Mathematical Theory of Evidence*. Princeton University Press, Princeton, NJ.

Shepherd, D. and Kumar, S. (2005). Microsensor applications. In Iyengar, S. S. and Brooks, R. R., editors, *Distributed Sensor Networks*, chapter 2, pages 11–27. Chapman & Hall/CRC.

Smarandache, F. and Dezert, J. (2005). Proportional conflict redistribution rules for information fusion. In *Proceedings of International Conference on Information Fusion 2005*.

Smarandache, F. and Dezert, J., editors (2006a). *Advances and Applications of DSmT for Information Fusion*, volume 2. American Research Press, Rehoboth.

Smarandache, F. and Dezert, J. (2006b). Proportional conflict redistribution rules for information fusion. In (Smarandache and Dezert, 2006a), chapter 1, pages 3–68.

Smets, P. (1990). The combination of evidence in the transferable belief model. *IEEE Pattern Analysis and Machine Intelligence*, 12:447–458.

Smets, P. (1993). Belief functions: the disjunctive rule of combination and the generalized Bayesian theorem. *International Journal of Approximate Reasoning*, 9:1–35.

Smets, P. (2000). Data fusion in the transferable belief model. In *Proceedings of International Conference on Information Fusion 2007*, volume 1, pages PS–21–PS–33.

Smets, P. (2005). Belief functions on real numbers. *International Journal of Approximate Reasoning*, 40(3):181–223.

Smets, P. (2007). Analysing the combination of conflicting belief functions. *Information Fusion*, 8(4):387–412.

Smets, P. and Kennes, R. (1994). The Transferable Belief Model. *Artificial Intelligence*, 66(2):191–234.

Smets, P. and Ristic, B. (2004). Kalman filter and joint tracking and classification in the TBM framework. In *Proceedings of International Conference on Information Fusion 2004*, pages 46–53.

Szewczyk, R., Mainwaring, A., Polastre, J., Anderson, J., and Culler, D. (2004). An analysis of a large scale habitat monitoring application. In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems*, pages 214–226, New York, NY, USA. ACM.

TinyOS Community (2010). TinyOS home page. `http://www.tinyos.net/` [Accessed 23 Oct 2010].

UC Berkeley and MLB Co. (2001). 29 palms fixed/mobile experiment. http://robotics.eecs.berkeley.edu/~pister/29Palms0103/ [Accessed 19 Oct 2010].

United States (1994). *Intelligence Preparation of the Battlefield: Field Manual 34-130.* Headquarters, Dept. of the Army, Washington, D.C.

Vanderbilt University (2008). Tracking of radio nodes. http://www.isis.vanderbilt.edu/projects/rips [Accessed 19 Oct 2010].

Voorbraak, F. (1990). A computationally efficient approximation of dempster-shafer theory. In Gaines, B. R. and Boose, J. H., editors, *Machine Learning and Uncertain Reasoning*, pages 461–472. Academic Press Ltd., London, UK.

Warneke, B., Last, M., Liebowitz, B., and Pister, K. S. J. (2001). Smart dust: Communicating with a cubic-millimeter computer. *Computer*, 34:44–51.

Welch, G. and Bishop, G. (2004). An introduction to the Kalman filter. From the Department Of Computer Science, University of North Carolina–Chapel Hill.

Whitehouse, C. D. (2002). The design of calamari: an ad-hoc localization system for sensor networks. Master's thesis, University of California, Berkeley.

Williams, J. L. (2007). *Information Theoretic Sensor Management.* PhD thesis, Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Williams, J. L., Fisher III, J. W., and Willsky, A. S. (2005a). An approximate dynamic programming approach for communication constrained inference. In *Proceedings of IEEE Workshop on Statistical Signal Processing.*

Williams, J. L., Fisher III, J. W., and Willsky, A. S. (2005b). An approximate dynamic programming approach to a communication constrained sensor management problem. In *Proceedings of International Conference on Information Fusion 2005.*

Williams, J. L., Fisher III, J. W., and Willsky, A. S. (2007). Approximate dynamic programming for communication-constrained sensor network management. *IEEE Transactions on Signal Processing*, 55(8):4300–4311.

Zhao, F., Shin, J., and Reich, J. (2002). Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing Magazine*, 19(2):61–72.

Appendix A

Condor Management Tool

A large number of simulations were required for the experiments conducted for this thesis. Managing all these simulations without automation would not have been feasible. A tool, which is discussed in this appendix, was created to automate the management of simulations which are run using a High-Throughput Computing (HTC) system.

The use of HTC presented an attractive option for running simulations as 4,470 successfully completed simulations were required for this thesis (900 for Chapter 4, 3,300 for Chapter 5, and 270 for Chapter 6). It would not be feasible to run this many simulations on a single desktop computer. A typical simulation could take between 30 minutes and 3 hours on a average desktop computer. Cardiff University's Condor network[1] provided a pre-configured HTC system that helped to minimise the preparation time required to run simulations and maximise throughput.

The remainder of this appendix briefly outlines aspects of Condor that are of particular relevance to the simulations that were run for this thesis, and presents the Condor Management Tool that was created to automate the management of these simulations.

A.1 Condor

Condor[2] is a HTC system that uses computers to execute programs, or *jobs* — these computers are typically idle, utilising CPU cycles that might otherwise be wasted. Non-interactive jobs execute on computers that are part of the Condor pool. Condor matches jobs with available computers depending on both the job's and the computer's requirements. The remainder of this section briefly describes the life of a typical Condor job that would be run for this thesis — it does not provide a general discussion of Condor jobs, or provide a more in-depth account of the life of a Condor job.

[1] `http://www.cardiff.ac.uk/arcca/services/condor/arcca-condor.html`
[2] `http://www.cs.wisc.edu/condor/`

Since the algorithms for this thesis were implemented in Matlab, the Matlab Compiler[3] was used to create a single executable suitable for running all of the simulations required for this thesis. This executable was packaged with all the data required (Including target trajectories, terrain maps, and sensor layouts) for any scenario and framework configuration. The Matlab Compiler Runtime (MCR) is required to run compiled Matlab executables. The packaged executable and MCR were placed on a network share accessible from all of the Condor machines. All of the Condor jobs that were run for this thesis used a small batch file (`'do.bat'`) to add the MCR to the path and download and run the executable when the job starts.

Ideally, a Condor job will execute on a machine and finish normally. A job can be interrupted, for example if the machine is no longer idle, which can result in a job being *evicted.* When a job is evicted, it is placed back into the submit queue to run again when a machine becomes available. If a job is running and the machine is no longer idle, the job will initially be paused. If the job becomes idle again within a set period of time, then the job will resume, otherwise the job is evicted. Unfortunately, compiled Matlab programs typically produce an error when resuming from a pause — resulting in an error which terminates execution of the job. Consequently, when jobs are paused because a machine is no longer idle, one of two outcomes will occur: the job is evicted and placed back into the submit queue, or the job terminates prematurely.

In both of the above cases it is typically more efficient to recover from the interrupted job, continuing from a previous point of execution, than to start the job from the beginning. In order to achieve this it is necessary to perform *checkpointing.* A *checkpoint* is a saved execution state that can be used to continue execution. Automatic checkpointing is not supported by Condor for Windows processes — necessitating a manual checkpointing scheme. All of the simulations for this thesis produce a checkpoint a regular intervals. The checkpoint is always created at the end of a time step, and contains everything required to continue the simulation at the next time step.

A.1.1 Job Submission

A job is submitted using a *submit file.* A submit file describes one or more jobs that form a *cluster* of jobs; All jobs within a cluster use the same executable (Condor Team, 2009a, p. 20). Condor submit files resemble Unix shell script files. An example of a submit file can be seen in Listing A.1 — this example shows the start of the file used to submit a group of jobs for Scenario A of Chapter 5.

[3] `http://www.mathworks.co.uk/products/compiler/`

```
1  universe = vanilla
2  rank    = ( 5*KFlops ) + Memory
3  submitdir = D:\matt\Octave_Jobs\ThesisChap2Scenario1Smaller
4  should_transfer_files = YES
5  when_to_transfer_output = ON_EXIT_OR_EVICT
6  executable = $(submitdir)\do.bat
7  image_size = 262144
8
9  # Start of job. Group = 3000, layout = 0
10 clus  = 3000
11 layout  = 0
12 num_particles  = 1500
13 max_time_steps = 1000
14 output  = job-out_$(clus)-$(layout).txt
15 error   = job-err_$(clus)-$(layout).txt
16 log     = job-log_$(clus)-$(layout).txt
17 res_file = result_$(clus)-$(layout).txt
18 sensorLayout  = smallerSensorMap_1-0.mat
19 scenario = ThesisChap2Scenario1SmallerT0125-0.mat
20 initFun  = initMonteCarloTrial_Chap2_Scenario1_T0125_Smaller
21 planAlg  = mit+N~5+dp~comm
22 classAlg = none+resampleFun~standardResampleParticles
23 priority = 7
24 checkpoint_file = checkpoint_3000-0.mat
25 arguments = $(sensorLayout) $(scenario) $(max_time_steps) $(num_particles) $(
       res_file) $(checkpoint_file) $(planAlg) $(classAlg) $(initFun)
26 queue
27 # End of job
28
29 # Start of job. Cluster = 3000, layout = 1
30 clus  = 3000
31 layout  = 1
32 num_particles  = 1500
33 max_time_steps = 1000
34 ...
```

Listing A.1: A sample Condor submit file from Chapter 5, words in bold are keywords. Only the start of the file is shown to save space.

A submit file provides details to Condor about the requirements, parameters, and policies of jobs. For example, the `rank` argument is used to specify how Condor should prioritise the selection of machines for this job. A submit file can state which *universe* should be used, all jobs for this thesis used the 'vanilla' universe, where normal executables can be executed without any modifications.

Each time the `queue` command is used without any arguments, a new job is submitted. For more information about submit files see the Condor manual (Condor Team, 2009a, pp. 19-62). When a job file is submitted, the log file for the job (specified in the submit file with the `log` argument) is updated to show that the job has been submitted — this is the first entry in the log file for this job. All of the Condor jobs run for this thesis used a separate log file for each job.

Once a job has been submitted, it is added to the job queue. The job queue contains all the jobs that a user has submitted that have not yet finished. Each job has various activity states, including `idle`, `running`, and `completed`. When a job is submitted, it usually enters the `idle` state. When a job is matched to a machine that is available to run jobs, it is executed on that machine and enters the `running` state. A successful Condor job completes execution on a single machine briefly enters the `completed` state before being removed from the Condor queue.

A.1.2 File Transfer

When using the Windows operating system, files are transferred using Condor's file transfer mechanism by default. The file specified by `executable` is transferred automatically; all other files that are required for a Condor job to run must either be explicitly stated (using `transfer_input_files`) or must be transferred by some other means. All of the jobs that were run for this thesis transferred a single executable batch file which connected to a network share for the required files.

Files are transferred back to the submit machine using the policy specified by `when_to_transfer_output`. All of the Condor jobs for this thesis used the policy `ON_EXIT_OR_EVICT` — this means that files are transferred when a job finishes or is evicted; the files are sent back to back to the submit machine and also the next machine in the Condor pool that will run the job (Condor Team, 2009a, p. 27). This policy is used to transfer files from jobs that are evicted to recover from the checkpoint that is manually created by the compiled Matlab code at regular intervals. It is not necessary to specify which files to transfer back to the submit machine — all new and modified files that are in the temporary directory used by Condor are transferred; any subdirectories are ignored (Condor Team, 2009a, p. 54).

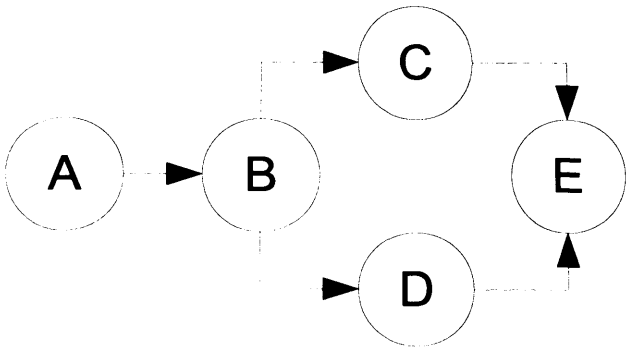


Figure A.1: An example DAG.

A.2 Job Management Tool — SimTools

The large number of Condor jobs that were run for this thesis necessitated the use of a job management automation tool. The remainder of this appendix describes 'SimTools' — the solution created to manage these jobs. SimTools can be seen as a Condor job meta-scheduler. Meta-schedulers, such as DAGMan (Condor Team, 2009a, pp. 89–116), manage jobs at a higher level than the Condor job scheduler (Condor Team, 2009b). SimTools allows the user to automate the simple tasks that are required to manage jobs — this includes writing submit files, collating results, and resubmitting failed jobs.

DAGMan would not have been suitable to manage the Condor jobs that were run for this thesis. DAGMan is suited to managing groups of jobs with inter-dependencies that form a larger task. A Directed Acyclic Graph (DAG) is used to specify job dependencies — for example five jobs, A–E, could be linked manner shown in Figure A.1; these jobs would be appropriate for management by DAGMan. The Condor jobs in this thesis are simulations that do not have inter-dependencies; it is only necessary that all jobs are successfully completed before the results can be analysed. A DAG is not an appropriate way to describe the simulations executed for this thesis. It is also unlikely that DAGMan would be able to investigate and resubmit failed jobs which appear to have completed execution successfully — a common occurrence with compiled Matlab programs.

SimTools was created specifically to manage compiled Matlab Condor jobs that run simulations for this thesis. It has the following features:

- Submit file creation
- Parameters are learnt automatically
- Results are collated

- Failed jobs are resubmitted
- Log output
- Provides an audit trail
- Runs at regular intervals
- Clean up after successful job completion

SimTools creates creates Condor submit files for a given set of simulations parameters; this can be used to either initially submit jobs manually or to automatically resubmit failed jobs. The parameters for the failed jobs can be remembered from the initial submits, but can also be parsed from result files of jobs that complete successfully — this provides the ability to both confirm parameters for already known scenarios and to learn the parameters for previously unknown scenarios. A new result file is created for each simulation; SimTools collates all of the results for a scenario into one file for easier analysis. Jobs that do not complete successfully are resubmitted — the job will recover from a checkpoint if it exists and the error is of a recognised type. The actions and findings of SimTools are saved to a new log file each time it is run (See Figure A.2 for an example). A snapshot, a cut-down checkpoint file, is taken at the end of a simulation, which combined with the SimTools log file, and the simulation log and error files provides sufficient information to investigate results. SimTools runs at regular intervals, typically every one to three hours, managing jobs at anti-social hours — when the largest number of machines are typically idle. To help save disk space, checkpoint files are deleted once SimTools has verified that a simulation has successfully completed.

A.2.1 Tasks

SimTools has two tasks that can be initiated by the user (or another calling program): create submit files, and process log and output files.

A.2.1.1 Create Submit Files

Submit files can be created by providing the following required parameters:

- Sensor layout string
- Sensor layout start number
- Sensor layout end number

```
06/11/2009,  6:20:00.89
Loading clusterParams from file (clusterParamsMap.dat)
Checkpoint file: checkpoint_5101-17.mat found and deleted as jobs is complete
Checkpoint file: checkpoint_5105-53.mat found and deleted as jobs is complete
Data file for job: 5107.96 not found, re-submitting
655 jobs still running
0 jobs need investigating
1 jobs to resubmit
Creating aggregate results file (C2S1_T025_Small_agResults_03_11_2009.txt), number of results: 244
Total number of jobs accounted for: 900
Saving clusterParams to file (D:\simTools\clusterParamsMap.dat)
Submitting job(s).
Logging submit event(s).
1 job(s) submitted to cluster 1604.
```

Figure A.2: Sample log out from SimTools. The log file is from Scenario A of Chapter 5.

- Submit filename
- Group number
- Planning algorithm
- Classification algorithm
- Number of particles
- Scenario string
- Maximum number of simulation time steps
- Initialisation function

`condor_submit` can then be executed with the submit filenames to submit for the initial submit to Condor. Sensor layout filenames consist of a sensor layout string, which is scenario specific, followed by a number, which is used to denote each unique sensor layout for a specific scenario, and then the file extension ('.mat'). For example, for 'scenarioXSensorLayout-' and '0' could be the sensor layout string and the sensor layout number, respectively for the 1st sensor layout of scenario X. The group number is the unique number that SimTools will use to refer to the set of simulations that are to be run using the specified parameters. The planning and classification algorithms are those described in Chapter 3. The number of particles to use in the particle filter must be specified. The scenario string is used in a similar way to the sensor layout string; when concatenated with the sensor layout number and '.mat' it forms the filename for the scenario file — the file that contains the target trajectory, and other required parameters such as the sampling interval. The initialisation function is used to initialise the data structures before the simulation begins.

A.2.1.2 Process Log and Output Files

The main usage of SimTools is to automate the management of Condor jobs — this is achieved by initiating the 'process' task. A Windows Scheduled task is used to run SimTools at regular intervals to perform job management. A high-level algorithm of this task can be found in Algorithm A.1 — the algorithm reads Condor log files, collates results, and resubmits jobs when necessary.

The main part of this task involves checking each log file to determine if the Condor job has completed (successfully or otherwise). This is achieved by checking the numbered event code at the start of the most recent log entry. If the job has not terminated

Algorithm A.1 Condor job management within SimTools.

```
for all logFilename ∈ logFilenames do {Each log filename}
  layoutNum = inferLayoutNum(logFilename)
  groupNum = inferGroupNum(logFilename)
  eventCode = getEventCode(logFilename) {Get most recent event code}
  if eventCode == TERMINATED then {Has job terminated}
    termCode = getTermCode(logFilename) {Parse log file for termination code}
    if termCode == 0 then {Check for successful termination code}
      resultFileName = getDataFileName(logFilename) {Infer from logFilename}
      success = processResultFile(resultFileName) {Parse result file, if the result
      file contains results then add them to the collated results and learn the param-
      eters for the group}
      if success == true then {Were results found and parsed from the result file?}
        checkPointFilename = inferCheckpointFilename(logFilename)
        if fileExists(checkPointFilename) == true then
          {Delete the checkpoint file if it exists as it is no longer required}
          delete(checkPointFilename)
        end if
      else {Results were not found, the result file only contained a header}
        {Resubmit job with checkpoint (if available)}
        addToResumeQueue(layoutNum,groupNum)
      end if
    else if termCode == COPY_ERROR or termCode == NETWORK_ERROR then
    {termCode indicates a 'normal' error}
      {Resume job from checkpoint (if available)}
      addToResumeQueue(layoutNum,groupNum)
    else {Job has failed for an abnormal reason}
      {Restart job (even if a checkpoint exists)}
      addToRestartQueue(layoutNum,groupNum)
    end if
  else if eventCode == ABORTED then {Has job been manually aborted?}
    {Resume job from checkpoint (if available)}
    addToResumeQueue(layoutNum,groupNum)
  else
    {Job has not terminated, assuming running or idle}
  end if
end for
a = length(resumeQueue)
b = length(resubmitQueue)
if a + b > 0 then
  Create submit file
  Submit jobs
end if
```

and has not been manually aborted, then it is assumed to be either running or idle and as such requires no attention from SimTools. If a job has terminated then the termination code is parsed to determine if the completed successfully or not. If a job is reported as completing successfully then it's result file is checked — if it contains a result then it is collated and the job's checkpoint file is deleted. A manually aborted job, or a recognised error will result in a job being resubmitted with a checkpoint. A job that has failed for any other reason will be restarted without a checkpoint file.

The process of checking log and output files is restarted each time the task is executed — the only information which is kept is the parameters for each group which are parsed from result files and remembered from submit tasks. Discarding the past knowledge of which jobs have completed and re-checking all the log files is more computationally expensive, but it is simpler and more robust. If for any reason it is necessary to re-execute a job (or set of jobs), it is as simple as aborting the job if it is idle or running, or deleting the checkpoint and result files if it has terminated. This configuration has also proved to be robust to computer restarts, which can be forced some Windows updates.